MASTER'S THESIS

Computational Thinking als Fundamentele Basisvaardigheid bij Programmeren

Een Onderzoek naar de Invloed van de Programmeeromgeving en Leerkrachtinterventies op de Ontwikkeling van Computational-Thinking-Vaardigheden bij Leerlingen in het Basisonderwijs.

Van Rossum, Desiré

Award date:
2020

MASTERTHESIS

Onderwijswetenschappen

Open Universiteit

Computational Thinking als Fundamentele Basisvaardigheid bij Programmeren

Een Onderzoek naar de Invloed van de Programmeeromgeving en Leerkrachtinterventies op de Ontwikkeling van Computational-Thinking-Vaardigheden bij Leerlingen in het Basisonderwijs

Computational Thinking: a Fundamental Skill

A Study on the Influence of the Programming Environment and Teacher Interventions on the Development of Computational Thinking Skills in Primary School Children

Door: Desiré van Rossum

Naam: Desiré van Rossum
E-mailadres: dvanrossum@xs4all.nl
Cursusnaam en cursuscode: Masterthesis OM9919172114
Naam begeleider: prof. Dr. Marcus Specht / Nardie Fanchamps MSc
Datum: 19-1-2020

Door het inleveren van dit voorstel verklaar ik dat het eigen werk is en dat het vrij is van plagiaat.

Inhoudsopgave

Computational thinking als fundamentele basisvaardigheid bij programmeren

Een onderzoek naar de invloed van de programmeeromgeving en leerkrachtinterventies op de ontwikkeling van computational-thinking-vaardigheden bij leerlingen in het basisonderwijs

Door: Desiré van Rossum

Samenvatting

Het belang van het aanleren van computational-thinking-vaardigheden aan jonge kinderen als één van de 21e-eeuwse vaardigheden wordt internationaal onderschreven. Computational thinking bestaat uit denkprocessen en vaardigheden waarmee complexe problemen opgelost kunnen worden met behulp van digitale tools. De bestaande richtlijnen blijken de leerkrachten te weinig houvast te bieden binnen het hedendaagse onderwijsprogramma om in de onderwijspraktijk op een effectieve wijze vorm te geven aan de ontwikkeling van computational-thinking-vaardigheden. Zowel visuele programmeeromgevingen in de vorm van computerprogramma's als unplugged-programmeeractiviteiten blijken geschikt te zijn voor het aanleren van computational-thinking-vaardigheden. Vooral constructivistische leeractiviteiten, die uitgaan van een actief leerproces, worden door leerkrachten als kansrijk gezien om computational-thinking-vaardigheden tijdens programmeerlessen aan te leren aan leerlingen in het basisonderwijs. De effectiviteit van de rol van het soort programmeeromgeving en de rol van de leerkracht in dit leerproces is echter nog weinig onderzocht.

Dit onderzoek had dan ook als doel om te onderzoeken wat de invloed van de gehanteerde programmeeromgeving is wanneer bovenbouwleerlingen in het basisonderwijs via programmeerlessen computational-thinking-vaardigheden ontwikkelen en welk effect de gehanteerde begeleidingsinterventie van de leerkracht daarbij heeft.

Daarvoor werden in dit onderzoek de deelnemende leerlingen over twee experimentele condities verdeeld, waarbij de ene helft unplugged-lessen aangeboden kreeg (verkregen van: Codekinderen.nl) en de andere helft plugged-lessen aangeboden kreeg (middels het online-computerprogramma Scratch). Via een pretest- en posttest-ontwerp werden de computational-thinking-vaardigheden van de leerlingen voorafgaand aan de interventie en na afloop van de interventie getest door middel van een test die door de onderzoeker zelf ontworpen is.

De AFL-Q (Pat-El, Tillema, Segers, & Vedder, 2013) werd gebruikt om de monitoring- en scaffolding-vaardigheden van de leerkrachten te meten. De versie voor leerkrachten (de TAFL-Q) werd door de leerkrachten ingevuld voorafgaand aan de interventie. De versie voor leerlingen (de SAFL-Q) werd na afloop van de interventie ingevuld door de leerlingen.

Via het netwerk van de onderzoeker werden drie reguliere basisscholen benaderd om deel te nemen aan het onderzoek. In totaal hebben 15 bovenbouwklassen deelgenomen aan het onderzoek, waarmee in totaal 15 leerkrachten en 268 leerlingen hebben deelgenomen aan het onderzoek.

Om de invloed van de gebruikte programmeeromgeving op de groei in computational-thinking-vaardigheden te toetsen is gebruik gemaakt van t-testen. Daarnaast zijn verschillende multipele-regressie-analyses uitgevoerd om de relaties tussen de variabelen te onderzoeken. Daaruit bleek dat de meeste groei in computational-thinking-vaardigheden plaatsvond bij leerlingen die de unplugged-lessen hadden gevolgd en bij leerlingen die les hadden gekregen van een leerkracht die zowel monitoring- als scaffolding-vaardigheden toepaste tijdens de lessen.

Op basis van het uitvoerde onderzoek zijn aanwijzingen gevonden dat unplugged-lessen voor leerkrachten met weinig ervaring in het geven van lessen in computational thinking een geschikte en effectieve vorm zijn om deze vaardigheden aan te leren aan leerlingen in de bovenbouw van het basisonderwijs. Daarbij verdient het de aanbeveling voor leerkrachten om tijdens deze lessen zowel monitoring-vaardigheden als scaffolding-vaardigheden in te zetten om de leerlingen zo optimaal mogelijk te begeleiden.

Keywords: computational thinking, unplugged, plugged, monitoring, scaffolding, programmeren, 21e-eeuwse vaardigheden

Computational Thinking: a Fundamental Skill

A Study on the Influence of the Programming Environment and Teacher Interventions on the Development of Computational Thinking Skills in Primary School Children

By: Desiré van Rossum

Summary

The importance of teaching computational thinking, one of the key 21st century skills, to (young) children has been internationally recognized. Computational thinking consists of thinking processes and skills that, with the assistance of digital tools, can be used to solve complex problems. Currently, the existing guidelines provided in many educational programs are insufficient to enable teachers to be able to effectively foster the development of computational thinking skills in their students. Both visual programming environments (i.e., computer programs) and unplugged programming activities are suitable for teaching computational thinking skills. In particular, many teachers see constructivist learning activities, which are based on an active learning process, as a promising means of teaching computational thinking skills to pupils during programming lessons in primary education. However, the influence of the type of programming environment and the role of the teacher on the effectiveness of the learning process has not been sufficiently investigated.

Against this background, the aim of this research was to investigate the influence of the programming environment used for programming lessons and the effect of teachers' supervision intervention on the development of computational thinking skills in primary school students in the upper grades of primary school. For this purpose, the participating students were divided into two experimental groups in this study: one group received three unplugged lessons (retrieved from Codekinderen.nl), and the other group received plugged-in lessons using the online computer program Scratch. A pretest and posttest design was adopted, and the computational thinking skills (concepts, procedures, and attitudes) of the students were tested before and after the intervention through a test designed by the researcher.

Assessment for Learning questionnaires (Pat-El et al., 2013) were used to measure the monitoring and scaffolding skills of the teachers: teachers completed the Teacher Assessment for Learning Questionnaire before the intervention, and students completed the Student Assessment for Learning Questionnaire after it.

On the basis of personal connections with staff members, the researcher approached three regular primary schools to be venues for the research, and a total of 15 primary education classes participated, resulting in the participation of 15 teachers and 268 students.

To test the differences between the two programming conditions, *t* tests were used. In addition, several multiple regression analyses were performed to investigate the relationships between the variables. The results showed that the most growth in computational thinking skills was evident in students who received the unplugged lessons and in students who received lessons from a teacher who applied both monitoring and scaffolding skills during the lessons.

For teachers with little experience leading lessons in computational thinking, unplugged lessons can be a suitable and effective means of teaching these skills to students in the upper grades of primary school. In addition, it is advisable that teachers use both monitoring and scaffolding skills during these lessons to provide the students with the best possible guidance.

Keywords: computational thinking, unplugged, plugged, monitoring, scaffolding, programming, 21st century skills

Inleiding

Probleemschets en Doel van het Onderzoek

Om kinderen beter voor te bereiden op de maatschappij van de toekomst is het van belang dat reeds op de basisschool aandacht besteed wordt aan het aanleren van "21st century skills" die kinderen in hun latere leven nodig zullen hebben (Mannila et al., 2014; Thijs, Fisser, & Van der Hoeven, 2014; Voogt, & Roblin, 2010). Uit onderzoek blijkt dat deze 21st century skills doorgaans geen vast onderdeel vormen van het huidige curriculum van het basisonderwijs in Nederland (Thijs et al., 2014). Tevens is gebleken dat het onderdeel digitale geletterdheid de minste aandacht krijgt (Thijs et al., 2014). Leraren hebben de intentie om aandacht te besteden aan digitale geletterdheid, maar zijn niet in voldoende mate toegerust om hier vorm aan te geven (Thijs et al., 2014; Voogt & Roblin, 2010).

Een van de basisvaardigheden op het gebied van digitale geletterdheid is computational thinking (KNAW, 2012; Thijs et al., 2014). Wing (2006) noemt computational thinking een fundamentele basisvaardigheid die ieder kind zou moeten leren, zodat het kind in de toekomst in voldoende mate deel kan nemen aan de snel digitaliserende samenleving. Onder computational thinking wordt verstaan: het oplossen van complexe problemen met behulp van computertechnologie (Lye, & Koh, 2014; Wing 2006). Programmeren is een voorbeeld van een activiteit waarbij complexe problemen met behulp van computertechnologie worden opgelost. Onderzoek wijst uit dat programmeren een positief effect kan hebben op het ontwikkelen van computational-thinking-vaardigheden (Atmatzidou, Demetriadis, & Nika, 2018; Pardamean, & Suparyanto, 2015; Voogt, Brand-Gruwel, & Van Strien, 2017).

Om computational-thinking-vaardigheden via ICT-programmeeromgevingen te leren, kan een keuze gemaakt worden uit een groot scala aan leeromgevingen. Met name omgevingen die visueel programmeren als uitgangspunt hebben, blijken voor het basisonderwijs geschikt te zijn. Deze omgevingen vergen door hun eenvoudige visuele karakter minder cognitieve belasting van jonge leerlingen, waardoor de leerlingen beter in staat zijn om computational-thinking-vaardigheden te ontwikkelen (Lye & Koh, 2014). Scratch is een dergelijke visuele omgeving. Scratch is een online-computerprogramma waarmee leerlingen op eenvoudige wijze leren programmeren (Brennan & Resnick, 2012; Kalelioglu & Gülbahar, 2014; Korkmaz, 2016). Daarnaast zijn diverse 'unplugged'-activiteiten geschikt om zonder computer of ander apparaat de basisconcepten van computational thinking aan te leren. Door deze activiteiten leren kinderen hoe een computer en/of een robot werkt aan de hand van spelmaterialen (Bell, Alexander, Freeman, & Grimley, 2009; Faber, Wierdsma, Doornbos, Van der Ven, & De Vette, 2017; Wohl, Porter, & Clinch, 2015*)*.

Het versterken van de computational-thinking-vaardigheden tijdens programmeerlessen vereist een faciliterende programmeeromgeving en aansturing en begeleiding door de leerkracht op een constructivistische wijze. Van de leerkracht wordt een andere aanpak gevraagd dan in de lessen (bijvoorbeeld rekenlessen of taallessen) die via het principe van directe instructie gegeven worden

(Atmatzidou et al., 2018; Breed, 2004; Lehrer, Lee, & Jeong, 1999; Lye & Koh, 2014; Sentance & Csizmadia, 2017).

Het is nog niet inzichtelijk welke soort programmeeromgeving (plugged of unplugged) het meest geschikt is om basisschoolleerlingen computational-thinking-vaardigheden aan te leren. Daarnaast is niet bekend welke begeleidings- en interventietechnieken (gehanteerd door de leerkracht) het meest effectief zijn. Het doel van dit onderzoek is dan ook om te onderzoeken wat de invloed van de gehanteerde programmeeromgeving is als bovenbouwleerlingen via programmeerlessen computational-thinking-vaardigheden ontwikkelen en welk effect de gehanteerde begeleidingsinterventie van de leerkracht daarbij heeft.

Theoretisch Kader

Computational thinking. Het gedachtegoed van computational thinking werd geïntroduceerd door Papert (1980). In het laboratorium van het Massachusetts Institute of Technology (MIT) ontwikkelde Papert de programmeertaal LOGO om aan te tonen dat computertechnologie het leren en denken van kinderen drastisch kan veranderen. Papert zag computers niet als methodes om het lesgeven efficiënter te maken, maar als middel om kinderen zelf kennis te laten construeren. De basis voor computational thinking was daarmee gelegd. Hoewel er inmiddels verschillende definities van de term computational thinking bestaan, is de definitie van Wing (2011) de meest gangbare. Zij stelt dat computational thinking bestaat uit denkprocessen waarbij complexe problemen vertaald worden in een abstracte vorm, zodat deze problemen opgelost kunnen worden door een informatieverwerkende machine. De informatieverwerkende machine waar Wing (2011) naar verwijst kan zowel een mens als een computer zijn.

Programmeren en computational thinking. Programmeren wordt door velen gezien als een middel om computational-thinking-vaardigheden aan te leren (Lye & Koh, 2014; Papert, 1980; Rees, García-Peñalvo, Jormanainen, Tuul, & Reimann, 2016). Volgens Wing (2008) gaat het bij computational thinking om het oplossen van problemen op een abstracte en automatische wijze, waarbij gebruik kan worden gemaakt van computertechnologie. In de operationele definitie van The International Society for Technology in Education (ISTE) en The Computer Science Teachers Association (CSTA) (ISTE & CSTA, 2011) worden abstractie en automatisering genoemd als essentiële onderdelen van het probleemoplossende proces van computational thinking. Programmeren is bij uitstek een activiteit waarbij de computer kan worden gebruikt om abstracte problemen op te lossen en waarin automatisering een belangrijke rol speelt (Papert, 1980; Voogt et al., 2017). Een programmeeromgeving die geschikt is om computational-thinking-vaardigheden aan te leren aan jonge kinderen is het programma Scratch (Brennan & Resnick, 2012; Kalelioglu & Gülbahar, 2014; Korkmaz, 2016). De programmeertaal LOGO van Papert (1980) stond aan de basis van Scratch. Beide werden ontwikkeld door het MIT (Resnick et al., 2009). Scratch is speciaal ontworpen voor de leeftijdscategorie 8 tot 16 jaar. Bij Scratch kunnen kinderen hun eigen game, animatie, verhaal of aanverwante multimedia leren programmeren door visueel programmeerbare blokken aan elkaar te

koppelen of variabelen te manipuleren. Alle gemaakte projecten kunnen worden gedeeld in een online-community.

Naast programmeren via een computerprogramma kunnen computational-thinking-vaardigheden aangeleerd worden middels andere activiteiten. In de zogenoemde 'unplugged'-activiteiten leren kinderen de basisprincipes van programmeren en de werking van een computer zonder gebruik te maken van computers of soortgelijke apparaten. Unplugged-activiteiten zijn activiteiten waarin de werking van een computer of een robot wordt nagebootst door middel van spel en materialen (de leerlingen kunnen elkaar bijvoorbeeld als robot aansturen door middel van codes of commando's op losse kaartjes). Deze unplugged-activiteiten zijn laagdrempelig in gebruik in de onderwijspraktijk (Bell et al., 2009; Faber et al., 2017). Onderzoek naar dergelijke activiteiten staat nog in de kinderschoenen, maar de eerste onderzoeksresultaten tonen volgens Wohl et al. (2015) aan dat unplugged-activiteiten effectief zijn om basisconcepten als algoritmes, logisch redeneren en debuggen aan jonge kinderen aan te leren.

Computational-thinking-dimensies. Brennan en Resnick (2012) hebben geanalyseerd hoe computational-thinking-vaardigheden ontwikkeld worden tijdens het programmeren. Zij beschrijven in hun driedimensionale model drie dimensies die samen een definitie vormen van computational thinking: computational concepts, computational practices en computational perspectives. Dit is in lijn met de drie dimensies die volgens Bayman en Mayer (1988) tot ontwikkeling komen tijdens programmeerlessen: syntactic knowledge, conceptual knowledge en strategic knowledge. Computational thinking vereist kennis van abstracte begrippen. Dit zijn de eerdergenoemde computational concepts oftewel syntactic knowledge. Daarnaast moet een persoon die wil programmeren deze abstracte begrippen kunnen toepassen. Dit zijn de computational practices oftewel de conceptual knowledge. Ten slotte vereist computational-thinking vaardigheden als het analyseren en oplossen van problemen, waarbij begrippen en toepassingen strategisch ingezet worden. Dit valt onder de laatste dimensie: computational perspectives oftewel strategic knowledge.

Zhong, Wang, Chen en Li (2016) gebruikten het driedimensionale model van Brennan en Resnick (2012) om een assessment uit te werken voor het meten van de computational-thinking-vaardigheden van leerlingen in het basisonderwijs en het voortgezet onderwijs. Het bleek mogelijk te zijn om met open ontwerpopdrachten alle drie dimensies van computational thinking te toetsen. In het assessment van Zhong et al. (2016) zijn de volgende concepten getoetst: objects, instructions, sequences, loops, parallelism, events, conditionals, operators en data. Voor de computational practices werden de volgende onderdelen in het assessment verwerkt: planning and designing, abstracting and modelling, modularizing and reusing, iterative and optimizing en testing and debugging. De computational perspectives die getoetst waren zijn: creative and expressing, communicating and collaborating en understanding and questioning.

Het SLO[1] (2018) heeft de negen concepten van de uitwerking van het ISTE en CSTA (2011) vertaald om een voorbeeld te schetsen van een leerlijn voor het Nederlandse onderwijs. Deze leerlijn is vervolgens door een aantal schoolbesturen in samenwerking met het SLO, de PO-raad en Kennisnet als basis gebruikt voor de leerlijn "Programmeren in het PO" (Stichting Kennisnet, 2016). De leerlijn is opgebouwd met behulp van de volgende begrippen: algoritmes, decompositie, patronen, herhaling, fouten, voorwaarden, abstractie, functie, variabele en representatie. Deze begrippen worden als volgt omschreven:

- Algoritmes zijn via instructie gevolgde vaste stappen die tot een vooraf vastgesteld doel leiden.
- Decompositie is het opdelen van een probleem in deelproblemen.
- Patronen zijn vaste herhalende structuren in bijvoorbeeld modellen, vorm of kleur.
- Herhaling wordt ook wel een 'lus' of een 'loop' genoemd.
- Fouten worden ook wel 'bugs' genoemd.
- Een voorwaarde is een regel waaraan voldaan moet worden voordat de volgende stap gezet kan worden.
- Abstractie is het algemeen maken van problemen door de verschillen weg te halen.
- Functie is een hulpprogramma binnen het hoofdprogramma.
- Een variabele is een waarde die kan variëren.
- Een representatie is een model waarin je de gegevens weergeeft.

Op basis van deze begrippen zijn doelen geformuleerd en worden in de leerlijn suggesties gedaan voor bruikbare activiteiten. Deze begrippen en doelen vertonen meerdere overeenkomsten met de concepten, de procedures en de attitudes uit het model van Brennan en Resnick (2012) en het assessment van Zhong et al. (2016). Om alle gewenste computational-thinking-vaardigheden in zijn totaliteit te ontwikkelen dienen de drie eerdergenoemde dimensies tijdens de lessen aan bod te komen.

Constructivisme en programmeren. Papert (1980) pleitte in zijn boek voor het aanleren van programmeren op een constructivistische wijze met behulp van ontdekkend leren. De constructivistische leertheorie gaat uit van een actief leerproces waarin kennis door de lerende zelf geconstrueerd wordt, al dan niet in een sociale context (Bruner, 1996; Dewey, 2007; Papert, 1980; Piaget, 2005; Vygotsky, 1980). Constructivisme komt voort uit de psychologische ontwikkelingstheorie van Piaget (2005). Daarbij ging Piaget uit van een kennistheorie waarin kennis wordt beschouwd als een construct dat door de mens gevormd is, als een afspraak tussen mensen onderling. Vygotsky (1980), Dewey (2007) en Bruner (1996) benadrukken de basiskenmerken van een constructivistische leeromgeving: een leeromgeving waarin kennis door de leerlingen actief geconstrueerd wordt in een sociale context en waarin voorkennis als vertrekpunt dient voor het

[1] Stichting Leerplan Ontwikkeling

doelgerichte leren. Dit vereist specifieke begeleiding van de leerkracht om de leerling actiever en doelgerichter te laten leren.

Het constructivisme heeft ook het wetenschapsgebied met betrekking tot computertechnologie en informatica beïnvloed (Ben-Ari, 1998). Onderzoek wijst uit dat interventies in programmeerlessen die via constructivistische principes worden vormgegeven positieve leeruitkomsten als resultaat hebben (Lye & Koh, 2014). Leerkrachten die ervaring hebben met het lesgeven in computational-thinking-vaardigheden benoemden in het onderzoek van Sentance en Csizmadia (2017) een aantal principes waarop zij hun leeractiviteiten gebaseerd hebben en die gezien kunnen worden als constructivistische leeractiviteiten. Dit zijn de volgende vaardigheden: actief leren, zelfontdekkend leren, probleemoplossend leren, ervaringsgericht leren en sociaal leren. Leerlinggestuurde leeractiviteiten (zoals probleemgestuurd leren, onderzoekend leren en ontdekkend leren) worden door leerkrachten vaker genoemd als activiteiten die bruikbaar zijn in lessen waarin computational-thinking-vaardigheden aangeleerd worden, in plaats van traditionele leerkrachtgestuurde leeractiviteiten (zoals directe instructie en modelling) (Bower et al., 2017).

Hoewel er duidelijke aanwijzingen zijn dat leerkrachten die programmeerlessen geven over inhoudelijke, didactische en vakdidactische kennis van computational thinking dienen te beschikken (Slangen, 2016; Thompson, Bell, Andreae, & Robins, 2013), is niet bekend welke begeleidings- en interventietechnieken tijdens de programmeerlessen het meest effectief zijn. Gezien de raakvlakken met constructivistische leeractiviteiten is het van belang om te onderzoeken welke leerkrachtinterventies en begeleidingstechnieken in dergelijke settings het meest effectief zijn, om vervolgens in kaart te brengen of deze effectiviteit ook optreedt in programmeerlessen. Hoewel constructivistische leeractiviteiten uitgaan van het principe van actieve kennisverwerving door de lerende zelf, is begeleiding door de leerkracht een belangrijke voorwaarde voor de effectiviteit van deze aanpak (Alfieri, Brooks, Aldrich, & Tenenbaum, 2011; Kirschner, Sweller, & Clark, 2006; Lazonder & Harmsen, 2016).

Effect van verschillende soorten begeleiding en interventie. In verschillende onderzoeken wordt een poging gedaan om een onderscheid te maken tussen de effecten van begeleidingstechnieken, instructietechnieken en interventietechnieken. Uit de review van Lazonder en Harmsen (2016) blijkt dat de verschillen in leeruitkomsten niet verklaard kunnen worden door verschillende begeleidingsvormen tijdens het onderzoekend leren. Lazonder en Harmsen (2016) maakten een onderscheid tussen meer specifiekere begeleidingsvormen (zoals scaffolding en uitleg geven) en minder specifieke begeleidingsvormen (zoals prompts of procesbegeleiding). Alfieri et al. (2011) vonden wel een verschil in leeruitkomsten bij ontdekkende leeractiviteiten die te verklaren waren door andere begeleidingstechnieken. Zo waren technieken als het geven van feedback, scaffolding en uitleg door de leerkracht effectieve technieken om het leerproces te ondersteunen. De volgende begeleidingstechnieken bleken nog effectiever te zijn: uitgewerkte voorbeelden geven, het leerproces

bij de leerlingen ondersteunen door de voorkennis van de leerlingen te vergroten en leerlingen de strategieën hardop laten verwoorden (Kirschner et al., 2006).

Scaffolding is een vorm van ondersteuning die afgebouwd kan worden naarmate een persoon zijn leerproces meer kan sturen (Wood, Bruner, & Ross, 1976). Het proces van scaffolding tijdens probleemoplossend leren is in detail onderzocht door Wood et al. (1976). In het onderzoek van deze auteurs gaf een effectieve tutor ondersteuning aan de lerende. De tutor zorgde door het stellen van goede vragen ervoor dat de lerende interesse hield in de taak, de taak vereenvoudigd werd, de lerende aangespoord werd tot doelgericht gedrag, de lerende op de belangrijkste concepten werd gewezen, de lerende niet door frustratie afhaakte en een voorbeeld werd gegeven. Dat dergelijke ondersteuning ook in een klassensetting gegeven kan worden, blijkt uit de casestudie van Fessakis, Gouli en Mavroudi (2013). In dit onderzoek gaf een leerkracht een klassikale les programmeren aan leerlingen in de leeftijd van 5 en 6 jaar. De leerkracht ondersteunde het leerproces door de leerlingen aan te moedigen, te zorgen voor succeservaringen en de interactie tussen leerlingen te bevorderen. Uit al deze onderzoeken blijkt dat de leerkracht een belangrijke rol speelt in het leerproces van de leerlingen.

Directe instructie blijkt een effectieve methode te zijn voor het aanleren van conceptuele basiskennis en het aanleren van vaste stappenplannen bij procedurele kennis (Kirschner et al., 2006). Als het echter gaat om het aanleren van generieke vaardigheden (zoals probleemoplossende vaardigheden en het vergroten van het inzicht van de leerlingen), dan is een constructivistische aanpak (zoals scaffolding) de beste methode voor zelfgestuurd leren (Choi, Lindquist, & Song, 2014).

Monitoring en scaffolding. Het is gebleken dat het leren van vaardigheden zonder begeleiding minder effectief is. De effecten van begeleidings- en interventietechnieken zijn echter beperkt onderzocht en de onderzoeken hiernaar leveren niet altijd eenduidige conclusies op (Hmelo-Silver, Duncan, & Chinn, 2007; Kirschner et al., 2006). Interventies dienen altijd op het leerproces van de lerende afgestemd te worden, waarbij scaffolding een veelvoorkomende strategie is (Hmelo-Silver et al., 2007). Om af te stemmen dient de leerkracht het leerproces van de leerlingen goed te monitoren. Monitoring en scaffolding zijn vaardigheden die een leerkracht nodig heeft om leerlingen de juiste feedback te geven en om de ontwikkeling van de leerlingen vanuit de zone van de naaste ontwikkeling te stimuleren (Heritage, 2007; Shepard, 2000). Via monitoring en scaffolding kan een leerkracht binnen een groep eenvoudiger differentiëren dan wanneer de groep in een instructie of in een presentatie als geheel aangesproken wordt (Heritage, 2007).

Vraagstellingen en Hypothesen

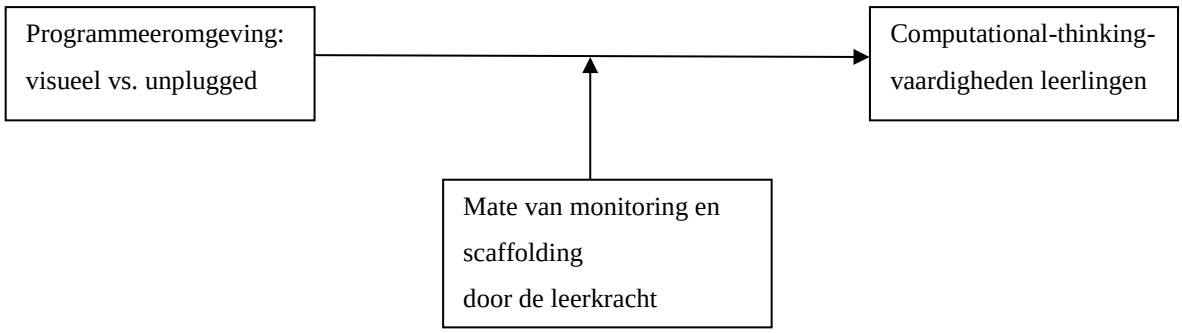
Uit de onderzoeken die tot op heden verricht zijn, is niet eenduidig naar voren gekomen welk soort programmeeromgeving (unplugged/plugged) het beste gebruikt kan worden om computational-thinking-vaardigheden aan te leren aan basisschoolleerlingen. Tevens is onvoldoende inzichtelijk in hoeverre monitoring en scaffolding als begeleidings- en interventietechnieken bijdragen aan het effectief aanleren van deze vaardigheden. De centrale onderzoeksvraag van dit onderzoek luidt daarom als volgt: *Wat is de invloed van de gehanteerde programmeeromgeving op de ontwikkeling van*

computational-thinking-vaardigheden van bovenbouwleerlingen tijdens programmeerlessen en welk effect hebben de door de leerkracht gehanteerde monitoring- en scaffolding-technieken op de ontwikkeling van deze vaardigheden?

Om deze hoofdvraag te beantwoorden, zijn de volgende deelvragen en bijbehorende hypothesen opgesteld:

1. *In hoeverre zorgen unplugged-programmeerlessen voor een hogere mate van ontwikkeling van computational-thinking-vaardigheden dan programmeerlessen die gegeven zijn met behulp van een visuele (plugged) programmeeromgeving?* De hypothese is dat unplugged-programmeerlessen voor meer groei in computational-thinking-vaardigheden zorgen dan de plugged-programmeerlessen.
2. *Zullen bij programmeerlessen die gegeven zijn door leerkrachten die vooraf aangeven de vaardigheden monitoring en scaffolding toe te passen tijdens hun dagelijkse lespraktijk de leerlingen zich beter ontwikkelen ten aanzien van computational-thinking-vaardigheden dan bij lessen die gegeven zijn door leerkrachten die vooraf aangeven monitoring en scaffolding minder toe te passen?* De hypothese is dat hoe vaker leerkrachten monitoring- en scaffolding-vaardigheden toepassen in de dagelijkse lespraktijk, hoe hoger de groei zal zijn in computational-thinking-vaardigheden bij de leerlingen.
3. *Heeft de mate van ingezette monitoring- en scaffolding-vaardigheden van de leerkracht tijdens de unplugged-activiteiten een grotere invloed op de ontwikkeling van de computational-thinking-vaardigheden dan bij de visuele (plugged) programmeeromgeving Scratch?* De hypothese is dat tijdens de unplugged-activiteiten dit een grotere invloed heeft dan tijdens de plugged-activiteiten.

De veronderstelde relaties tussen de variabelen zullen volgens bovenstaande hypotheses schematisch als volgt eruitzien:



Figuur 1: Schematische weergave van de veronderstelde relaties tussen de variabelen

Methode

Ontwerp

Om de veronderstelde relaties uit de hypotheses te onderzoeken is in dit onderzoek een exploratief ontwerp gehanteerd waarin twee experimentele condities vergeleken werden. In dit onderzoek is geen controlegroep gehanteerd. Om de deelnemende leerlingen qua geslacht en qua cognitief niveau evenredig over de twee condities te verdelen werden de leerlingen per klas eerst gematcht, om de leerlingen vervolgens willekeurig te verdelen over de twee condities. Het cognitieve niveau van de leerlingen werd bepaald aan de hand van het resultaat van de laatste afname van Cito LVS rekenen. De ene groep leerlingen kreeg drie unplugged-programmeerlessen van 45 minuten en de andere groep kreeg drie plugged-programmeerlessen van 45 minuten (met behulp van het computerprogramma Scratch). De ene groep kreeg les van de eigen leerkracht. De andere helft van de groep kreeg tegelijkertijd les van de onderzoeker in een andere ruimte. Om ervoor te zorgen dat de onderzoeker niet bij alle klassen enkel de unplugged of de plugged lessen gaf, wisselde de onderzoeker hierin per klas af.

Daarnaast is het onderzoek uitgevoerd middels een pretest/posttest-ontwerp. De ontwikkeling van de computational-thinking-vaardigheden van de leerlingen werd voorafgaand aan de interventie en na afloop van de interventie gemeten door middel van een zelf ontworpen toets, waarmee de drie computational-thinking-dimensies (Brennan & Resnick, 2012; Zhong et al., 2016) getoetst werden als een geheel (zie bijlage 1 en 2). Voorafgaand aan de interventie is gemeten in welke mate de leerkracht monitoring- en scaffolding-vaardigheden toepast in de dagelijkse lespraktijk. Dit is gemeten door de afname van een vragenlijst. Na afloop van de interventie hebben de leerlingen de leerlingenversie van deze vragenlijst ingevuld, om in kaart te brengen in hoeverre de leerkrachten tijdens de programmeerlessen monitoring- en scaffolding-vaardigheden hebben toegepast. In tabel 1 is het ontwerp van het onderzoek schematisch weergegeven.

Tabel 1

Beschrijving van het onderzoeksdesign

Verdeling over groepen		**Onderzoeksactiviteiten**		
Willekeurige toewijzing na matching (binnen een klas)	Experimentele groep 1	Pretest: meting CT-vaardigheden en vragenlijst monitoring/scaffolding door leerkracht	3 unplugged-programmeerlessen	Posttest: meting CT-vaardigheden en vragenlijst monitoring/scaffolding door leerlingen
Willekeurige toewijzing na matching (binnen een klas)	Experimentele groep 2	Pretest: meting CT-vaardigheden en vragenlijst monitoring/scaffolding door leerkracht	3 programmeerlessen met behulp van Scratch	Posttest: meting CT-vaardigheden en vragenlijst monitoring/scaffolding door leerlingen

Participanten

Het onderzoek is uitgevoerd op drie reguliere basisscholen. De eerste school die benaderd is, is basisschool de Hazesprong te Nijmegen waar de onderzoeker zelf werkzaam is. Acht klassen van deze school hebben deelgenomen aan dit onderzoek: twee groepen 5, een groep 6, een groep 7, twee groepen 8 en twee Leonardoklassen (groep 5/6 en groep 7/8).

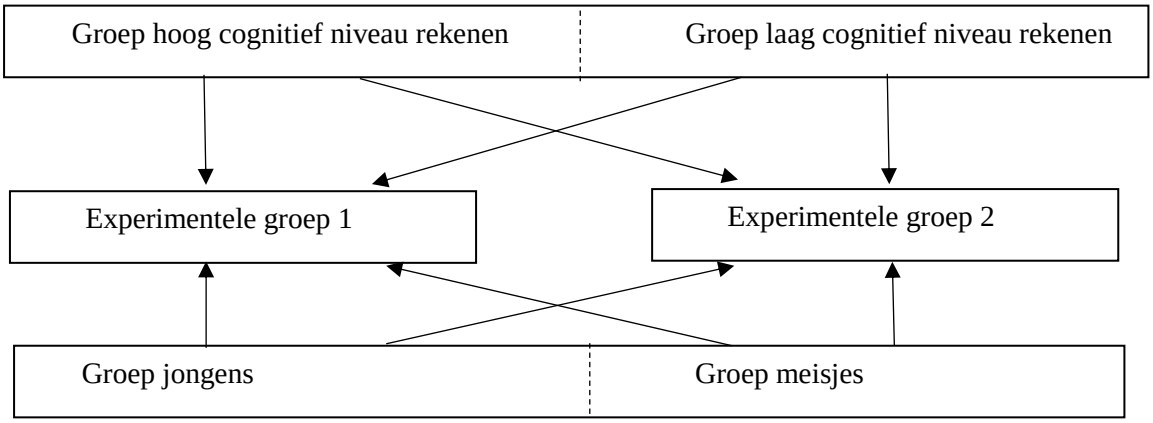
Daarnaast hebben twee andere basisscholen deelgenomen aan het onderzoek. De tweede school is basisschool de Viersprong te Wijchen. Alle vier bovenbouwklassen van deze school hebben deelgenomen aan dit onderzoek: een groep 5, een groep 6, een groep 7 en een groep 8. De derde school die heeft deelgenomen aan dit onderzoek is basisschool Brakkenstein te Nijmegen. Drie van de vijf bovenbouwgroepen van deze school hebben deelgenomen aan dit onderzoek: een groep 5, een groep 6 en een groep 7.

In totaal hebben vijftien bovenbouwklassen van drie reguliere basisscholen deelgenomen aan dit onderzoek: vier groepen 5, een groep 5/6, drie groepen 6, drie groepen 7, een groep 7/8 en drie groepen 8. De steekproef bestond uit 79 leerlingen uit groep 5 (8-jarigen en 9-jarigen), 58 leerlingen uit groep 6 (9-jarigen en 10-jarigen), 62 leerlingen uit groep 7 (10-jarigen en 11-jarigen) en 69 leerlingen uit groep 8 (11-jarigen en 12-jarigen). Dat waren in totaal 268 leerlingen. Per deelnemende klas heeft één leerkracht deelgenomen aan het onderzoek (in totaal 15 leerkrachten).

Iedere deelnemende klas werd verdeeld over de twee experimentele groepen. De eigen leerkracht had inspraak in de verdeling van de groepen. De jongens en de meisjes werden apart verdeeld over de twee condities. Ook werden de leerlingen met een hoger en lager cognitief niveau op het gebied van rekenen apart verdeeld over de twee condities. Deze indeling is gemaakt op basis van de score bij de laatste toets van het Cito LVS (score in de laagste 50% = lager cognitief niveau, score in de hoogste

50% = hoger cognitief niveau). Deze werkwijze is gehanteerd om de invloed van het geslacht en het cognitief niveau te kunnen controleren.

Schematisch ziet de verdeling van de leerlingen over de twee experimentele condities er als volgt uit:



Figuur 2. Schema van de verdeling van de leerlingen over de twee experimentele groepen

In het overzicht in bijlage 3 is weergegeven hoe de leerlingen qua aantallen over de experimentele condities zijn verdeeld.

Materialen

Om de computational-thinking-vaardigheden van leerlingen te meten, dienden de leerlingen tien programmeeropdrachten uit te voeren zonder computer. In deze opdrachten moesten de leerlingen een Lego-poppetje met behulp van commando's door een doolhof leiden. Aangezien nog geen expliciet en gevalideerd meetinstrument beschikbaar is om computational-thinking-vaardigheden te meten, zijn de opdrachten in deze test door de onderzoeker zelf ontworpen met behulp van de online-lessuggestie op de website http://www.tacle3.eu/nederlands/2017/03/22/een-lego-doolhof-maken/. Bij het ontwerp van deze opdracht is uitgegaan van de computational-thinking-dimensies volgens Brennan en Resnick (2012). De eerste helft van de opdrachten bestond uit open programmeeropdrachten. De tweede helft van de opdrachten had betrekking op het opsporen en verbeteren van bestaande programmeerfouten. Per opdracht konden de leerlingen maximaal 2 punten verdienen. In totaal konden de leerlingen maximaal 20 punten verdienen voor deze test. Een toelichting bij deze opdrachten is te vinden in bijlage 1.

De mate van monitoring en scaffolding door de leerkracht is in kaart gebracht door middel van de 'Assessment for Learning Questionnaires' (AFL-Q) (Pat-El et al., 2013). Dit is een vragenlijst met 28 vragen. Van de vragenlijst bestaan twee versies: een versie voor leerlingen (SAFL-Q) en een versie voor leerkrachten (TAFL-Q) (zie bijlage 4 en 5). De betrouwbaarheid van deze vragenlijsten is volgens de ontwerpers van de vragenlijst goed (bij de TAFL-Q is monitoring α = .87 en scaffolding α = .77 en bij de SAFL-Q is monitoring α = .89 en scaffolding α = .83).

Beide vragenlijsten bestaan uit 28 stellingen. Per stelling kan op een 5-punts-Likertschaal aangegeven worden in hoeverre de genoemde activiteit door de leerkracht gehanteerd wordt tijdens de lessen. Antwoordopties variëren van 'vrijwel nooit' (optie 1) tot en met 'met grote regelmaat' (optie 5).

Een voorbeeldstelling uit de leerkrachtlijst luidt: *Ik laat mijn leerlingen weten wat hun sterke punten zijn op het gebied van leren.* Dezelfde stelling ziet er op de leerlingenlijst als volgt uit: *Mijn leerkracht vertelt tegen mij wat mijn sterke punten zijn op het gebied van leren.*

De variabelen monitoring en scaffolding werden op ordinaal niveau gemeten. De variabele monitoring werd gemeten aan de hand van 16 items uit de vragenlijst (voor ieder item score 1-5), waardoor deze totaalscore uitgedrukt kan worden in een waarde tussen 16 en 80. De variabele scaffolding werd gemeten aan de hand van 12 items uit de vragenlijst (voor ieder item score 1-5), waardoor deze totaalscore uitgedrukt kan worden in een waarde tussen 12 en 60.

De vragenlijst werd voorafgaand aan de interventie aan de leerkrachten voorgelegd, om in kaart te brengen in hoeverre de leerkrachten in hun dagelijkse lespraktijk reeds in staat zijn om monitoring- en scaffolding-technieken toe te passen. Na afloop van de interventie werd de vragenlijst ingevuld door de leerlingen om de mate van toegepaste monitoring- en scaffolding-technieken tijdens de programmeerlessen in kaart te brengen.

Er is een betrouwbaarheidsanalyse uitgevoerd met behulp van Cronbach's Alpha. In dit onderzoek was de betrouwbaarheid van de TAFL-Q vragenlijst voor monitoring $\alpha = .813$ en voor scaffolding $\alpha = .835$. Daarnaast was de betrouwbaarheid van de SAFL-Q vragenlijst voor monitoring $\alpha = .889$ en voor scaffolding $\alpha = .821$. Daarmee zijn de vragenlijsten voldoende betrouwbaar gebleken.

Als visuele programmeeromgeving is het programma Scratch gebruikt. Dit is een gratis online-programma waarin kinderen een project ontwerpen (bijvoorbeeld een spel of een animatie) met behulp van eenvoudige visuele codes (zie: https://scratch.mit.edu/). Voor de unplugged-activiteiten werd gebruik gemaakt van de lessen van Codekinderen (zie: https://maken.wikiwijs.nl/100525/CodeKinderen#!page-3210948). De leerkrachten hebben de lessen 'leer robottaal', 'schrijf je naam binair' en 'lego codetaal' aan de leerlingen gegeven.

Procedure

Nadat op 27 november 2018 de cETO[2] formeel goedkeuring had gegeven voor de uitvoering van dit onderzoek, is het onderzoek als eerste gestart bij basisschool de Hazesprong in Nijmegen. In een teambijeenkomst in februari 2019 zijn de opzet, het doel en het verloop van het onderzoek uitgelegd en is aan de leerkrachten toestemming gevraagd om deel te nemen aan het onderzoek. Bij een parttime aanstelling kon slechts één van de leerkrachten deelnemen aan het onderzoek. Tevens is aan de directie van de school toestemming gevraagd voor deelname aan dit onderzoek.

[2] cETO = commissie ethische toetsing onderzoek van de Open Universiteit.

Voor de start van de lessen zijn aan alle kinderen een informatiebrief en een toestemmingsbrief meegegeven. Alle kinderen mochten meedoen met de programmeerlessen. In het onderzoek werden echter alleen de gegevens verzameld van de kinderen die het door hun ouders getekende toestemmingsformulier hadden ingeleverd.

Nadat zowel de leerkrachten als de directie het toestemmingsformulier hadden ingevuld, hebben de leerkrachten tijdens de teambijeenkomst individueel de TAFL-Q ingevuld na een korte toelichting van de onderzoeker. Daarna hebben alle leerkrachten de Wikiwijs-workshop Computational Thinking (ontworpen door Kennisnet) gevolgd. De leerkrachten hebben deze workshop gevolgd om ervoor te zorgen dat alle leerkrachten basiskennis hadden van computational thinking voordat zij de lessen zouden geven. De workshop werd door de leerkrachten op een ander moment voortgezet, maar werd uiterlijk voor de start van het onderzoek in de klas afgerond. De leerkrachten kregen tevens heldere instructies over het verloop van het onderzoek, zodat zij tussentijds geen andere lesactiviteiten zouden ondernemen die het onderzoek konden beïnvloeden.

Voor iedere klas werden in de periode maart tot en met mei 2019 drie momenten van 45 minuten ingepland waarop de programmeerlessen gegeven zouden worden. De leerkracht gaf alle drie de lessen aan één experimentele groep. De onderzoeker gaf drie lessen aan de andere experimentele groep. Om ervoor te zorgen dat de onderzoeker niet bij alle klassen enkel de unplugged of de plugged lessen gaf, wisselde de onderzoeker hierin per klas af. Hierdoor kon bij de data-analyse de verschillen tussen de unplugged en de plugged conditie vergeleken worden en werd dit niet beïnvloed doordat de onderzoeker slechts één conditie les had gegeven.

Voorafgaand aan de interventie werd het niveau van computational-thinking-vaardigheden van alle leerlingen gemeten door middel van een aantal opdrachten in een test. Bij het maken van deze test was alleen de onderzoeker aanwezig in de klas. De leerkracht was hierbij niet aanwezig in de klas. Deze werkwijze is toegepast om te garanderen dat de leerkracht niet wist hoe de leerlingen getest zouden worden, zodat dit niet de wijze van lesgeven kon beïnvloeden.

Na de drie gegeven lessen vond een nameting plaats van de computational-thinking-vaardigheden van de leerlingen. Bij deze nameting is wederom gebruik gemaakt van dezelfde test. Ten slotte vulden alle leerlingen individueel de SAFL-Q in, om het leerkrachtgedrag tijdens de programmeerlessen in kaart te brengen.

Na de start van het onderzoek op de eerste school werden twee andere scholen benaderd via het netwerk van de onderzoeker. In de periode april 2019 tot en met juni 2019 werd het onderzoek gelijktijdig uitgevoerd op basisschool de Viersprong in Wijchen en op basisschool Brakkenstein in Nijmegen.

Data-analyse

Voor het testen van de eerste hypothese zijn de scores van de twee experimentele groepen vergeleken door middel van een onafhankelijke t-test, waarbij groei in computational-thinking-vaardigheden de afhankelijke variabele was en de twee omgevingen de onafhankelijke variabelen. Om tot een score van

de groei in computational-thinking-vaardigheden te komen, is per leerling het verschil berekend tussen de score op de posttest en de score op de pretest. Voorafgaand aan deze analyse is door middel van een gepaarde t-test berekend of er een significant verschil was tussen de totaalscore op de pretest en de totaalscore op de posttest.

Voor het testen van de tweede en derde hypothese is gebruik gemaakt van meerdere multipele-regressie-analyses. Om de tweede hypothese te testen, werden de relaties onderzocht tussen de groei in computational-thinking-vaardigheden van de leerlingen tijdens de programmeerlessen en de monitoring- en scaffolding-vaardigheden van de leerkracht in de dagelijkse lespraktijk. Om de derde hypothese te testen, werden de relaties onderzocht tussen de groei in computational-thinking-vaardigheden bij de leerlingen en de toegepaste monitoring- en scaffolding-vaardigheden van de leerkracht (achteraf ingevuld door de leerlingen). Hierbij werden de separate invloed van de programmeeromgeving en de mate van monitoring en scaffolding op de computational-thinking-vaardigheden berekend, evenals het interactie-effect en/of het modererende effect van de variabelen. De onderzoeker heeft in dit onderzoek in alle groepen les gegeven en heeft de verschillende lessen meermaals gegeven. De eigen leerkracht heeft tijdens dit onderzoek enkel één serie van drie lessen gegeven. De onderzoeker had meer voorkennis van computational thinking dan de deelnemende leerkrachten. Om deze reden zijn de resultaten van deze twee groepen apart van elkaar geanalyseerd.

Bij alle analyses is gebruik gemaakt van het computerprogramma SPSS versie 26.0 en is een significantieniveau van 5% gehanteerd ($p < .05$).

Resultaten

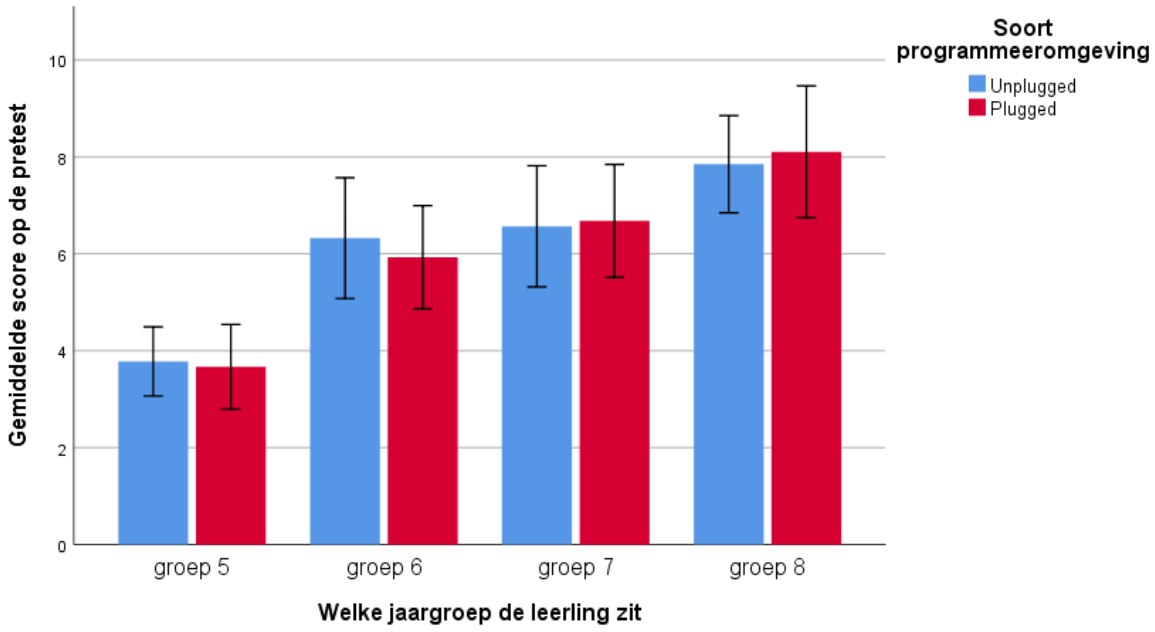
Pretest en Posttest Resultaten

Tijdens de afname van de pretest waren niet alle leerlingen die deel zouden nemen aan het onderzoek aanwezig. In totaal hebben 251 leerlingen de pretest gemaakt, zie tabel 2. De gemiddelde totaalscores van iedere jaargroep namen toe naarmate de jaargroep hoger werd.

Tabel 2
Score op de pretest

Jaargroep van de leerling	Soort programmeeromgeving	Gemiddelde score	N	s
Groep 5	Unplugged	3.78	36	2.11
	Plugged	3.67	36	2.59
	Totaal	3.72	72	2.35
Groep 6	Unplugged	6.32	31	3.40
	Plugged	5.93	28	2.75
	Totaal	6.14	59	3.09
Groep 7	Unplugged	6.57	30	3.35
	Plugged	6.68	28	3.01
	Totaal	6.62	58	3.16
Groep 8	Unplugged	7.85	33	2.83
	Plugged	8.10	29	3.58
	Totaal	7.97	62	3.18
Totaal	Unplugged	6.06	130	3.28
	Plugged	5.95	121	3.39
	Totaal	6.01	251	3.32

In figuur 3 zijn per jaargroep en per conditie de gemiddelde scores op de pretest weergegeven. In dit figuur is tevens zichtbaar dat de gemiddelde scores van de twee experimentele condities toenamen naarmate de jaargroep hoger werd. Daarnaast verschilden de twee experimentele groepen (unplugged; $N = 130$, $M = 6.06$, plugged; $N = 121$, $M = 5.95$) niet significant van elkaar ten aanzien van de score op de pretest (t (249) = .264, $p = .792$).



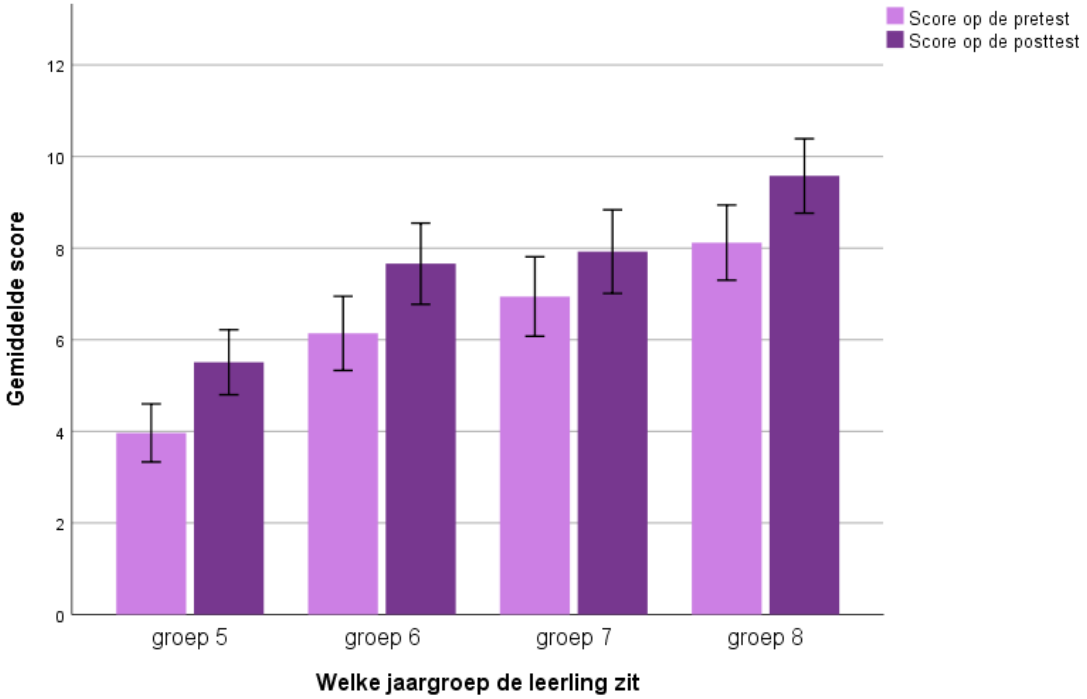
Figuur 3: Score op de pretest per jaargroep inclusief 95% betrouwbaarheidsintervallen

In tabel 3 staan alle gemiddelde scores van de posttest per jaargroep en experimentele conditie. In totaal hebben 232 leerlingen de posttest gemaakt.

Tabel 3
Posttest scores per jaargroep

Jaargroep van de leerling	Soort programmeeromgeving	Gemiddelde score	N	s
Groep 5	Unplugged	5.40	30	2.53
	Plugged	5.43	30	2.80
	Totaal	5.42	60	2.64
Groep 6	Unplugged	8.20	25	3.25
	Plugged	7.12	25	2.95
	Totaal	7.66	50	3.12
Groep 7	Unplugged	8.04	27	3.49
	Plugged	7.34	29	3.32
	Totaal	7.68	56	3.39
Groep 8	Unplugged	9.21	34	3.23
	Plugged	9.50	32	3.21
	Totaal	9.35	66	3.20
Totaal	Unplugged	7.73	116	3.42
	Plugged	7.40	116	3.39
	Totaal	7.56	232	3.40

In figuur 4 zijn vervolgens per jaargroep de gemiddelde scores op de pretest en de posttest naast elkaar weergegeven. Te zien is dat in alle groepen de scores op de posttest hoger zijn dan de scores op de pretest. In totaal is een significant verschil (*M* = 1,38) tussen de gemiddelde totaalscore van de pretest en de gemiddelde totaalscore van de posttest (*t* (217) = 7,42, *p* < .05).



Figuur 4: Verschil in gemiddelde scores op zowel de pretest als de posttest per jaargroep inclusief 95% betrouwbaarheidsinterval

Resultaten Deelvraag 1

In hoeverre zorgen unplugged-programmeerlessen voor een hogere mate van ontwikkeling van computational-thinking-vaardigheden dan programmeerlessen die gegeven zijn met behulp van een visuele (plugged) programmeeromgeving?

In tabel 4 staan per experimentele groep (unplugged, plugged) de gemiddelde scores vermeld van de groei in computational-thinking-vaardigheden. De leerlingen in de unplugged-conditie (*M* = 1.49) hebben beter gescoord dan de leerlingen in de plugged-conditie (*M* = 1.26). Dit verschil is echter niet significant (*t* (213) = 0.59, *p* =.557). De hypothese dat unplugged-programmeerlessen voor meer groei in computational-thinking-vaardigheden zorgen dan de plugged-programmeerlessen is daarmee niet bevestigd. Echter zijn de scores tevens gesplitst in de scores van de leerlingen die les hebben gekregen van de onderzoeker en de scores van de leerlingen die les hebben gekregen van hun eigen leerkracht. Bij de leerlingen die les hebben gekregen van de onderzoeker is er meer groei te zien voor de leerlingen die de plugged-lessen hebben gevolgd (*M* = 1,95) dan bij de leerlingen die de unplugged-lessen hebben gevolgd (*M* = 1,10). Dit verschil is niet significant (*t* (103) = - 1.52, *p* = .13). Bij de leerlingen die les hebben gehad van hun eigen leerkracht is er meer groei te zien bij de unplugged-

groep (*M* = 1,80) dan bij de plugged-groep (*M* = 0,47). Dit verschil is significant (*t*(107) = 2.75, *p* < .05). Voor de leerlingen die les hebben gehad van hun eigen leerkracht is daarmee wel aangetoond dat unplugged-programmeerlessen voor meer groei in computational-thinking-vaardigheden zorgen.

Tabel 4
Groei in computational-thinking-vaardigheden van onderzoeker en eigen leerkracht

Les gekregen door leerkracht of door onderzoeker	Soort programmeeromgeving	Gemiddelde score	*N*	*s*
Onderzoeker	Unplugged	1.10	49	2.50
	Plugged	1.95	57	3.23
	Totaal	1.56	106	2.93
	t (103) = -1.52 *p* = .133 [-1.95, 0.26]			
Eigen leerkracht	Unplugged	1.80	60	2.65
	Plugged	0.47	49	2.33
	Totaal	1.20	109	2.58
	t (107) = 2.75 *p* <.05 [0.37, 2.29]			
Totaal	Unplugged	1.49	109	2.59
	Plugged	1.26	106	2.93
	Totaal	1.38	215	2.76
	t (213) = 0.59 *p* = .557 [-0.52, 0.97]			

Resultaten Deelvraag 2

Zullen bij programmeerlessen die gegeven zijn door leerkrachten die vooraf aangeven de vaardigheden monitoring en scaffolding toe te passen tijdens hun dagelijkse lespraktijk de leerlingen zich beter ontwikkelen ten aanzien van computational-thinking-vaardigheden dan bij lessen die gegeven zijn door leerkrachten die vooraf aangeven monitoring en scaffolding minder toe te passen?

Tabel 5 bevat de totaalscores van de vragenlijsten (TAFL-Q) over de monitoring- en scaffolding-vaardigheden die door de leerkrachten zijn ingevuld. Tevens staan in de laatste kolom de gemiddelde scores van de groei in computational-thinking-vaardigheden van de groep leerlingen die les hebben gehad van hun eigen leerkracht. Middels de Shapiro-Wilk test (Field, 2013) is nagegaan of de scores afwijken van een normaalverdeling. De scores van de monitoring-vaardigheden van de leerkrachten (*D* (15) = 0.947, *p* = .484) weken niet significant af van een normaalverdeling. Ook de scores van de scaffolding-vaardigheden van de leerkrachten (*D* (15) = 0.935, *p* = .320) weken niet significant af van een normaalverdeling. Tot slot weken ook de scores van de groei in computational-thinking-vaardigheden van de groep leerlingen (*D* (15) = 0.922, *p* = .210) niet significant af van een normaalverdeling.

Tabel 5
Monitoring- en scaffolding-vaardigheden van de leerkrachten en groei CT leerlingen

Nummer groep	Monitoring-totaalscore volgens leerkracht	Scaffolding-totaalscore volgens leerkracht	Gemiddelde groei in CT van leerlingen
1	47.00	36.00	1.22
2	51.00	43.00	2.00
3	48.00	42.00	1.70
4	65.00	48.00	0.40
5	56.00	52.00	-0.90
6	56.00	50.00	1.20
7	70.00	54.00	0.00
8	56.00	43.00	1.88
9	56.00	47.00	-0.25
10	52.00	48.00	2.00
11	59.00	52.00	2.33
12	57.00	48.00	1.67
13	63.00	52.00	2.20
14	56.00	50.00	2.60
15	65.00	44.00	0.38
Totaal	57.13	47.27	1.23
	D (15) = 0.947 p = .484	D (15) = 0.935 p = .320	D (15) = 0.922 p = .210

Om de oorzaak-gevolgrelaties tussen monitoring en scaffolding en de groei in computational-thinking-vaardigheden bij de leerlingen te onderzoeken is een multipele-regressie-analyse uitgevoerd (op basis van de door de leerkrachten ingevulde vragenlijsten, namelijk TAFL-Q). Om de uitkomsten van de berekeningen te vergelijken zijn de gestandaardiseerde waardes van de variabelen gebruikt.

In tabel 6 is te zien dat de correlaties van de variabelen monitoring (r = -0.32, p = .134), scaffolding (r = -0.11, p = .360) en interactie monitoring/scaffolding (r = -0.03, p = .453) met de groei in computational-thinking-vaardigheden niet significant zijn (p > .05). Het berekende model middels de multipele regressie is ook niet significant (F(3,13) = 0.43, p = .739). Bij leerkrachten die vooraf aangegeven hebben de vaardigheden monitoring en scaffolding vaker toe te passen tijdens hun dagelijkse lespraktijk blijken de leerlingen zich niet beter te ontwikkelen ten aanzien van computational-thinking-vaardigheden dan bij lessen die gegeven zijn door leerkrachten die aangegeven hebben deze vaardigheden minder in te zetten in hun dagelijkse lespraktijk.

Tabel 6
Resultaten multipele-regressie-vragenlijsten leerkrachten

Model		Correlatie met groei CT *r*	Regressie-coëfficiënt *B*	*t*	*p*
1	(Constante)		0.05	0.15	.881
	Zscore (MonitoringLKR)	-0.32 (*p* = .134)	-0.41	-1.05	.320
	Zscore (ScaffoldingLKR)	-0.11 (*p* = .360)	0.12	0.30	.773
	ZmonscafLKR	-0.03 (*p* = .453)	-0.03	-0.13	.903

R^2 = 0.11
F (3,13) = 0.43 (*p* = .739)

Resultaten Deelvraag 3

Heeft de mate van ingezette monitoring- en scaffolding-vaardigheden van de leerkracht tijdens de unplugged-activiteiten een grotere invloed op de ontwikkeling van de computational-thinking-vaardigheden dan bij de visuele (plugged) programmeeromgeving Scratch?

Om de invloed van de toegepaste monitoring- en scaffolding-vaardigheden door de leerkracht in combinatie met de programmeeromgeving op de groei in computational-thinking-vaardigheden bij de leerlingen te onderzoeken, is tevens een multipele-regressie-analyse uitgevoerd (op basis van de door de leerlingen ingevulde vragenlijsten, namelijk SAFL-Q). In deze analyse zijn alleen de resultaten opgenomen van de leerlingen die les hebben gehad van hun eigen leerkracht. Om de uitkomsten van de berekeningen te vergelijken, zijn de gestandaardiseerde waardes van de variabelen gebruikt.

Als eerste stap is een multipele regressie uitgevoerd waarbij bovengenoemde variabelen individueel zijn ingevoerd in het model. In tabel 7 staan de uitkomsten van deze multipele regressie vermeld. Het model is niet significant ($F(3,99) = 2.57$, $p = .059$) en verklaart slechts 7% van de variantie in de groei van computational-thinking-vaardigheden. De correlatie van de soort programmeeromgeving met de groei in computational-thinking-vaardigheden is echter significant ($r = -0.24$, $p = .008$), evenals de regressie-coëfficiënt van de programmeeromgeving in dit model ($B = -0.40$, $t(103) = -2.26$, $p = .026$).

Tabel 7
Resultaten multipele regressie van individuele variabelen

Model		Correlatie met groei CT *r*	Regressie-coëfficiënt *B*	*t*	*p*
1	(Constante)		-0.09	-1.01	.315
	Omgeving (plugged)	-0.24 (*p* = .008)	-0.40	-2.26	.026
	Zscore (MonitoringLL)	0.15 (*p* = .071)	0.13	1.27	.208
	Zscore (ScaffoldingLL)	0.02 (*p* = .420)	-0.06	-0.54	.592

R^2 = 0.07
F (3,102) = 2.57 (*p* = .059)

Omdat de correlatie tussen monitoring en scaffolding als variabelen onderling (*r* = 0.48, *p* < .05) significant is, is in een volgende stap een multipele regressie uitgevoerd met monitoring en scaffolding als toegevoegde interactie-variabele. In tabel 8 zijn de resultaten van deze analyse vermeld. Het model is significant (*F*(2.102) = 8.88, *p* < .05) en verklaart 15% van de variantie in de score van de groei in computational-thinking-vaardigheden. De regressiecoëfficiënten van de omgeving (*B* = - 0.37, *t*(103) = -2.20, *p* = .030) en de interactievariabele (*B* = 0.26, *t*(103) = 3.32, *p* = .001) leveren in dit model een significante bijdrage.

Tabel 8
Resultaten multipele regressie met monitoring en scaffolding als interactie-variabele

Model		Correlatie met groeiCT *r*	Regressie-coëfficiënt *B*	*t*	*p*
2	(Constante)		- 0.19	- 2.14	.035
	Omgeving (plugged)	-0.24 (*p* =.008)	- 0.37	- 2.20	.030
	interactiemonitoringscaffoldingLL	0.33 (*p* < .05)	0.26	3.32	.001

R^2 = 0.15
F (2,102) = 8.88 (*p* < .05)

Het meenemen van monitoring en scaffolding als een interactievariabele verklaart de grotere variantie in de score van de groei in computational-thinking-vaardigheden. Om te onderzoeken in hoeverre monitoring en scaffolding tezamen een modererende rol spelen in de invloed van de programmeeromgeving, is ten slotte nogmaals een multipele-regressie-analyse uitgevoerd. In deze analyse is de interactie tussen de omgeving en de interactievariabele monitoring en scaffolding opgenomen als extra interactie-variabele. In tabel 9 staan de resultaten van deze analyse vermeld.

Hoewel ook dit model significant is (*F*(3.102) = 5.88, *p* <.05), is de verklaarde variantie in dit model (15%) gelijk aan de verklaarde variantie in het vorige model. Het valt op dat de nieuwe

interactie-variabele niet significant bijdraagt aan het model (r = -0.04, t(103) = -0.24, p = .811). Zelfs de invloed van de programmeeromgeving is in dit model niet meer significant (r = -0.36, t(103) = -1.95, p = .054).

Tabel 9
Resultaten multipele regressie als moderatie-analyse

Model		Correlatie met groei CT *r*	Regressie-coëfficiënt *B*	*t*	*p*
3	(Constante)		-0.19	-2.12	.036
	Omgeving (plugged)	-0.24 (p = .008)	-0.36	-1.95	.054
	interactiemonitoringscaffoldingLL	0.33 (p < .05)	0.25	3.05	.003
	interactieOmgMonScafLL	-0.20 (p = .020)	-0.04	-0.24	.811

R^2 = 0.15
F (3,102) = 5.88 (p < .05)

Uit de multipele-regressie-analyses blijkt dat model 2 het model is dat de meeste variantie verklaart in de groei in computational-thinking-vaardigheden, waaraan beide onafhankelijke variabelen (de omgeving en de interactie tussen monitoring en scaffolding) een significante bijdrage leveren. De ingezette monitoring- en scaffolding-vaardigheden moeten door de leerkracht dus tezamen ingezet worden en niet als afzonderlijke vaardigheden om een positief effect te hebben op de ontwikkeling van de computational-thinking-vaardigheden bij de leerlingen.

Conclusie en Discussie

Hoewel het belang van het aanleren van computational-thinking-vaardigheden internationaal wordt onderschreven als een van de 21e-eeuwse vaardigheden, heeft eerder onderzoek niet onomstotelijk kunnen uitwijzen welk soort programmeeromgeving (unplugged/plugged) het meest geschikt is om computational-thinking-vaardigheden bij basisschoolleerlingen aan te leren. Daarnaast is niet inzichtelijk welke begeleidings- en interventietechnieken de leerkracht het beste kan inzetten om deze vaardigheden op een effectieve wijze aan te leren. Dit rechtvaardigt de hoofdonderzoeksvraag van dit onderzoek: *Wat is de invloed van de gehanteerde programmeeromgeving op de ontwikkeling van computational-thinking-vaardigheden van bovenbouwleerlingen tijdens programmeerlessen en welk effect hebben de door de leerkracht gehanteerde monitoring- en scaffolding-technieken op de ontwikkeling van deze vaardigheden?* Om deze onderzoeksvraag te beantwoorden zijn drie deelvragen met bijbehorende hypotheses geformuleerd en getoetst.

Conclusie Deelvraag 1

In hoeverre zorgen unplugged-programmeerlessen voor een hogere mate van ontwikkeling van computational-thinking-vaardigheden dan programmeerlessen die gegeven zijn met behulp van een visuele (plugged) programmeeromgeving? In totaal was er geen significant verschil in groei in computational-thinking-vaardigheden tussen de leerlingen die de unplugged-lessen hebben gevolgd en de leerlingen die de plugged-lessen hebben gevolgd. Hiermee kon de hypothese dat unplugged-programmeerlessen voor meer groei in computational-thinking-vaardigheden zorgen dan de plugged-programmeerlessen niet bevestigd worden. Echter was bij de groep leerlingen die enkel les hadden gehad van hun eigen leerkracht dit verschil wel significant. De hypothese is daarmee deels bevestigd. Bij de leerlingen die les hebben gehad van de onderzoeker waren de computational-thinking-vaardigheden van de plugged-groep sterker gegroeid.

Het verschil in ervaring met het geven van computational-thinking-lessen kan het verschil in de onderzoeksresultaten verklaren. Het geven van unplugged-lessen is voor leerkrachten laagdrempelig en is vergelijkbaar met reguliere lessen waarin leerlingen in groepjes ontdekkend en/of onderzoekend leren (Brackmann et al., 2017; Bell et al., 2009; Faber et al., 2017). Voor leerkrachten zijn unplugged activiteiten een passend analoog alternatief wanneer er op school geen technologische middelen beschikbaar zijn (Humble, Mozelius & Sällvin, 2019). Bij de plugged-lessen is ervoor gekozen om met een specifiek computerprogramma te werken (namelijk Scratch) dat wellicht meer inhoudelijke kennis van programmeren vereist. Er zijn duidelijke aanwijzingen dat leerkrachten die programmeerlessen geven inhoudelijke, didactische en vakdidactische kennis nodig hebben van computational thinking (Slangen, 2016; Thompson, Bell, Andreae, & Robins, 2013). Voor de unplugged-lessen is dit minder relevant (Brackmann et al., 2017). Hoewel er nog weinig onderzoek is gedaan naar het verschil tussen het effect van verschillende programmeeromgevingen, wijst onderzoek uit dat zowel unplugged-activiteiten als plugged-activiteiten geschikt zijn om computational-thinking-concepten aan te leren (Humble et al., 2019; Lye & Koh, 2014; Wohl et al., 2015).

Conclusie Deelvraag 2

Zullen bij programmeerlessen die gegeven zijn door leerkrachten die vooraf aangeven de vaardigheden monitoring en scaffolding toe te passen tijdens hun dagelijkse lespraktijk de leerlingen zich beter ontwikkelen ten aanzien van computational-thinking-vaardigheden dan bij lessen die gegeven zijn door leerkrachten die vooraf aangeven monitoring en scaffolding minder toe te passen? In dit onderzoek zijn geen significante relaties gevonden tussen de mate waarin leerkrachten in hun dagelijkse lespraktijk monitoring- en scaffolding-technieken toepassen en de ontwikkeling van computational-thinking-vaardigheden bij de leerlingen. De hypothese dat hoe vaker leerkrachten monitoring- en scaffolding-vaardigheden toepassen in de dagelijkse lespraktijk, hoe hoger de groei zal zijn in computational-thinking-vaardigheden bij de leerlingen, is daarmee niet bevestigd. Aan dit onderzoek hebben relatief weinig leerkrachten (15) deelgenomen. Wellicht was wel een significante relatie gevonden indien meer leerkrachten betrokken waren geweest bij dit onderzoek. Dit lag echter buiten de mogelijkheden van dit onderzoek. Een andere mogelijke verklaring voor dit resultaat houdt in dat er geen relatie bestaat tussen de vaardigheden die de leerkrachten doorgaans inzetten in hun lespraktijk en de vaardigheden die de leerkrachten inzetten tijdens computational-thinking-lessen. De meeste basisschoolleerkrachten in Nederland voelen zich niet vertrouwd met het geven van programmeerlessen en het aanleren van computational thinking (Slangen, 2016; Thijs et al., 2014). Het is mogelijk dat de leerkrachten tijdens de programmeerlessen andere vaardigheden hebben ingezet dan dat ze normaal gesproken zouden doen in hun dagelijkse lespraktijk.

Conclusie Deelvraag 3

Heeft de mate van ingezette monitoring- en scaffolding-vaardigheden van de leerkracht tijdens de unplugged-activiteiten een grotere invloed op de ontwikkeling van de computational-thinking-vaardigheden dan bij de visuele (plugged) programmeeromgeving Scratch? Zowel de programmeeromgeving als de toegepaste monitoring- en scaffolding-vaardigheden van de leerkrachten hadden apart van elkaar invloed op de groei in computational-thinking-vaardigheden bij de leerlingen. De unplugged lessen die gegeven zijn door de deelnemende leerkrachten zorgden voor meer groei in computational-thinking-vaardigheden bij de leerlingen dan de plugged lessen. Dit is in lijn met de bevindingen bij de eerste deelvraag. De toegepaste monitoring- en scaffolding-vaardigheden van de leerkrachten waren alleen als gecombineerde variabele positief van invloed op de ontwikkeling van de computational-thinking-vaardigheden bij de leerlingen. Monitoring- en scaffolding technieken blijken nauw met elkaar samen te hangen (Pat-El et al., 2013). Er is geen modererende rol gevonden voor de monitoring- en scaffolding-variabele op de programmeeromgeving. De hypothese dat tijdens de unplugged-activiteiten de mate van ingezette monitoring- en scaffolding-vaardigheden een grotere invloed op de ontwikkeling van computational-thinking-vaardigheden bij de leerlingen heeft dan tijdens de plugged activiteiten, is daarmee niet bevestigd.

Conclusie Hoofdonderzoeksvraag

Het doel van dit onderzoek was om te onderzoeken wat de invloed van de gehanteerde programmeeromgeving is wanneer bovenbouwleerlingen via programmeerlessen computational-thinking-vaardigheden ontwikkelen en welk effect de gehanteerde begeleidingsinterventie van de leerkracht daarbij heeft. Samengevat kan worden gesteld dat de meeste groei in computational-thinking-vaardigheden bij leerlingen plaatsvond wanneer zij unplugged-les kregen van hun eigen leerkracht of wanneer zij plugged-les kregen van de onderzoeker. Zowel unplugged- als plugged-activiteiten bleken effectief te zijn om computational-thinking-vaardigheden bij basisschoolleerlingen aan te leren. Wanneer een leerkracht nog weinig voorkennis heeft van computational thinking en wanneer er weinig technologische middelen beschikbaar zijn op school lijken vooral unplugged-actviteiten geschikt om computational-thinking-vaardigheden aan te leren (Brackmann et al., 2017; Humble et al., 2019). De inhoudelijke, didactische en vakdidactische kennis van leerkrachten kan echter bijgeschoold worden (Slangen, 2016), waardoor ook de plugged-activiteiten effectief ingezet kunnen worden op de basisschool. Een gecombineerde aanpak wordt daarom aanbevolen (Bell & Vahrenhold, 2018).

Leerlingen ontwikkelden zich beter in computational-thinking-vaardigheden wanneer de leerkracht zowel monitoring- als scaffolding-vaardigheden toepaste tijdens de lessen. Het gegeven dat monitoring- en scaffolding-leerkrachtvaardigheden zijn die effectief ingezet kunnen worden om leerlingen de juiste feedback te geven in de klassenpraktijk is bekend uit eerder onderzoek (Heritage, 2007; Shepard, 2000). Of deze vaardigheden ook specifiek effectief zijn tijdens programmeerlessen is tot op heden nog niet eerder onderzocht. Leerkrachten kunnen geschoold worden in de wijze waarop zij monitoring- en scaffolding-technieken inzetten tijdens programmeerlessen (Slangen, 2016).

Implicaties

De resultaten van dit onderzoek moeten met enige voorzichtigheid geïnterpreteerd worden. De toegepaste mate van monitoring en scaffolding van de leerkrachten is onderzocht door vragenlijsten in te laten vullen door de leerlingen. Tijdens het onderzoek bleek dat het vooral voor de jongere leerlingen (in de leeftijd van 8 en 9 jaar) moeilijk was om de taal in de vragenlijsten te begrijpen. Als het taalgebruik in de vragenlijsten eenvoudiger was geweest, hadden de leerlingen mogelijk beter aan kunnen geven in hoeverre de vaardigheden daadwerkelijk door de leerkracht werden toegepast tijdens de lessen. Wellicht is een vragenlijst een te complexe wijze van dataverzameling en zou een andere wijze van dataverzameling een betrouwbaardere manier zijn om data te verzamelen. Een voorbeeld hiervan is het interviewen van de leerlingen of directe observatie van het gedrag van de leerkrachten tijdens de lessen.

Daarnaast werd zowel in de toetsing van de computational-thinking-vaardigheden (namelijk in de vorm van een opdracht op papier) als in de unplugged-lessen geen computerprogramma gebruikt. Dit raakvlak kan de resultaten van de leerlingen die de unplugged-lessen hebben gevolgd positief hebben beïnvloed (Brackmann et al., 2017). In vervolgonderzoek is het van belang om te onderzoeken in

hoeverre dezelfde resultaten behaald worden als op een andere wijze (bijvoorbeeld digitaal) getoetst wordt.

Ten tijde van dit onderzoek was er nog geen betrouwbare en valide test voorhanden waarmee alle dimensies (concepten, procedures en attitudes) van computational-thinking-vaardigheden tegelijkertijd gemeten konden worden. In andere onderzoeken werden vooral testen gebruikt die enkel een deel van deze karakteristieken in kaart brachten (Brackmann et al., 2017; Román-González, Pérez-González, & Jiménez-Fernández, 2017; Zhong et al., 2016). Om deze reden is in dit onderzoek ervoor gekozen om zelf een test te ontwikkelen op basis van de basisconcepten volgens Brennan en Resnick (2012). Op de ontworpen test haalden oudere leerlingen een hogere score dan de jongere leerlingen. Dit is een eerste aanwijzing voor de betrouwbaarheid van de ontworpen test. De metacognitieve ontwikkeling van oudere leerlingen is doorgaans verder gevorderd, waardoor zij op een hoger abstract niveau kunnen denken (Kuhn, 1999). Hierdoor kunnen oudere leerlingen dus een hogere score op de test halen dan de jongere leerlingen. Echter is de betrouwbaarheid en validiteit van de zelfontworpen test niet nader onderzocht. Om onderzoek naar computational-thinking-vaardigheden verder uit te breiden, is een betrouwbare en gevalideerde test nodig. Daarmee kunnen vervolgens specifieke interventies onderzocht worden, waardoor in kaart gebracht kan worden hoe leerkrachten computational-thinking-vaardigheden in de praktijk het beste kunnen aanleren en hoe zij dit leerproces het beste kunnen ondersteunen. Om dit te realiseren dient de ontworpen test in vervolgonderzoek gevalideerd te worden en dienen de gegevens die hieruit voortkomen nader geanalyseerd te worden.

Hoewel het doen van praktijkonderzoek tijdrovend kan zijn en moeilijker te controleren is dan in een experimentele setting, is in dit onderzoek er toch voor gekozen om de uitvoering en de dataverzameling specifiek toe te passen in de klassenpraktijk. Lessen werden soms korter door omstandigheden (zoals een schoolfotograaf die tussendoor bij de klas langskwam). Niet alle leerlingen waren altijd aanwezig tijdens de ingeplande momenten van het onderzoek en de activiteiten van het onderzoek moesten precies ingepast worden in het reguliere curriculum. Dit kostte veel aanpassingsvermogen en flexibiliteit van de onderzoeker. Daarnaast was het in verband met de beperkte tijd van het onderzoek niet mogelijk om meer dan drie lessen in te plannen. Wellicht had een uitbreiding van het aantal lessen een ander resultaat opgeleverd.

Leerkrachten hebben behoefte aan concrete suggesties waarin wordt beschreven hoe zij computational-thinking-lessen kunnen geven die effectief zijn en waarvan de effectiviteit is bewezen in praktijkonderzoek (Thijs et al., 2014; Voogt & Roblin, 2010). Om deze reden is in dit onderzoek toch ervoor gekozen om onderzoek te doen in de onderwijspraktijk, ondanks de beperkingen hiervan.

Referenties

Alfieri, L., Brooks, P. J., Aldrich, N. J., & Tenenbaum, H. R. (2011). Does discovery-based instruction enhance learning? *Journal of Educational Psychology*, *103*(1), 1-18.

Atmatzidou, S., Demetriadis, S., & Nika, P. (2018). How does the degree of guidance support students' metacognitive and problem solving skills in educational robotics? *Journal of Science Education and Technology*, *27*(1), 70-85.

Bayman, P., & Mayer, R. E. (1988). Using conceptual models to teach BASIC computer programming. *Journal of Educational Psychology*, *80*(3), 291-298.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20-29.

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work?. In *Adventures Between Lower Bounds and Higher Altitudes* (pp. 497-521). Springer, Cham.

Ben-Ari, M. (1998). Constructivism in computer science education. *ACM SIGCSE Bulletin*, *30*(1), 257-261.

Bower, M., Wood, L. N., Lai, J. W., Howe, C., Lister, R., Mason, R., ... Veal, J. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, *42*(3), 53-72.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 65-72). ACM.

Breed, B. (2003). The reflective abilities of expert and novice learners in computer programming. In *The British Educational Research Association Annual Conference* (pp. 6–14). Edinburgh, Verenigd Koninkrijk: Heriot-Watt University.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association* (p. 25). Vancouver: Canada: AERA.

Bruner, J. (1996). *Towards a theory of instruction*. Cambridge, MA: Harvard University Press.

Choi, E., Lindquist, R., & Song, Y. (2014). Effects of problem-based learning vs. traditional lecture on Korean nursing students' critical thinking, problem-solving, and self-directed learning. *Nurse education today, 34*(1), 52-56.

Dewey, J. (2007). *Experience and education*. New York, NY: Simon and Schuster.

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., Van der Ven, J. S., & de Vette, K. (2017). Teaching computational thinking to primary school students via unplugged programming lessons. *Journal of the European Teacher Education Network, 12*, 13-24.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97.

Field, A. (2013). *Discovering statistics using IBM SPSS statistics.* Thousand Oaks, CA: Sage.

Heritage, M. (2007). Formative assessment: What do teachers need to know and do? *Phi Delta Kappan, 89*(2), 140-145.

Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem based and inquiry learning: A response to Kirschner, Sweller, and Clark. *Educational psychologist, 42*(2), 99-107.

Humble, N., Mozelius, P., & Sällvin, L. (2019). On the Role of Unplugged Programming in K-12 Education. In *European Conference on e-Learning 2019 (ECEL2019)* (pp. 224-230). Academic Conferences and Publishing International Limited.

ISTE & CSTA. (2011). *Operational definition of computational thinking for K-12 education.* Geraadpleegd van http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinking Flyer.pdf

Kalelioglu, F., & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics In Education, 13*(1), 33-50.

Stichting Kennisnet. (2016). *Programmeren in het PO*. Geraadpleegd van https://maken.wikiwijs.nl/74282/Programmeren_in_het_PO

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75-86.

KNAW. (2012). *Digitale geletterdheid in het voortgezet onderwijs: Vaardigheden en attitudes voor de 21ste eeuw*. Geraadpleegd van https://www.knaw.nl/nl/actueel/publicaties/digitale-geletterdheid-in-het-voortgezet-onderwijs/@@download/pdf_file/20121027.pdf

Korkmaz, Ö. (2016). The Effect of Scratch- and Lego Mindstorms Ev3-Based Programming Activities on Academic Achievement, Problem-Solving Skills and Logical-Mathematical Thinking Skills of Students. *Malaysian Online Journal Of Educational Sciences, 4*(3), 73-88.

Kuhn, D. (1999). A developmental model of critical thinking. *Educational Researcher, 28*(2), 16-46.

Lazonder, A. W., & Harmsen, R. (2016). Meta-analysis of inquiry-based learning: Effects of guidance. *Review of Educational Research, 86*(3), 681-718.

Lehrer, R., Lee, M., & Jeong, A. (1999). Reflective teaching of logo. *The journal of the Learning Sciences, 8*(2), 245-289.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (pp. 1-29). New York, NY: ACM.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Pardamean, B., & Suparyanto, T. (2015). Improving Problem-Solving Skills through Logo Programming Language. *The New Educational Review, 41*(3), 52-65.

Pat-El, R. J., Tillema, H., Segers, M., & Vedder, P. (2013). Validation of assessment for learning questionnaires for teachers and students. *British Journal of Educational Psychology, 83*(1), 98-113.

Piaget, J. (2005). *The psychology of intelligence*. London, United Kingdom: Routledge.

Rees, A., García-Peñalvo, F. J., Jormanainen, I., Tuul, M., & Reimann, D. (2016). *An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers*. doi:10.5281/zenodo.165123

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. B. (2009). Scratch: Programming for all. *Commun. Acm, 52*(11), 60-67.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678-691.

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies, 22*(2), 469-495.

Shepard, L. A. (2005). Linking formative assessment to scaffolding. *Educational Leadership, 63*(3), 66-70.

Slangen, L. (2016). *Teaching robotics in primary school*. Geraadpleegd van https://pure.tue.nl/ws/files/25754482/20160630_CO_Slangen.pdf

SLO. (2018). *Leerlijnen Digitale Geletterdheid*. Geraadpleegd van https://slo.nl/vakportalen/vakportaal-digitale-geletterdheid/leerlijnen-digitale-geletterdheid/

Thijs, A., Fisser, P., & Van der Hoeven, M. (2014). *21e eeuwse vaardigheden in het curriculum van het funderend onderwijs*. Geraadpleegd van http://downloads.slo.nl/Repository/21e-eeuwse-vaardigheden-conceptueel-kader.pdf

Thompson, D., Bell, T., Andreae, P., & Robins, A. (2013). The role of teachers in implementing curriculum changes. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 245-250). New York, NY: ACM.

Voogt, J., & Roblin, N. P. (2010). *21st century skills: Discussienota.* Geraadpleegd van https://www.21stcenturyskills.nl/download/21_st_century_skills__UT_discussie_paperNL.pdf

Voogt, J., Brand-Gruwel, S., & Van Strien, J. (2017). *Effecten van programmeeronderwijs op computational thinking: een reviewstudie*. Geraadpleegd van https://www.nro.nl/wp-content/uploads/2017/05/003-Antwoord-Rapport-Programmeeronderwijs.pdf

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard university press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences, 366*(1881), 3717-3725.

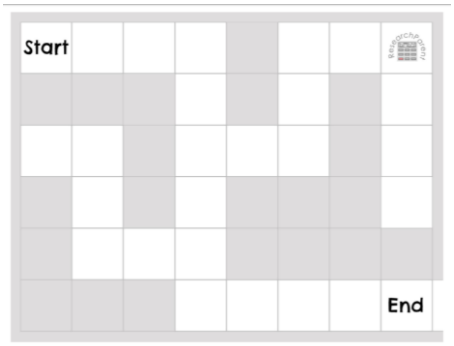
Wing, J. M. (2011). *Research notebook: computational thinking-what and why? The Link*. Geraadpleegd van http://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 55-60). New York, NY: ACM.

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, *17*(2), 89-100.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research, 53*(4), 562-590.

Bijlage 1: Opdrachten test Computational-thinking-vaardigheden



Opdracht 1:

Benodigdheden:

- 1 poppetje/pion
- Blad met codeermogelijkheden
- Doolhof nummer 1
- Deze opdrachtkaart met daarbij het invulblad

De opdracht:
Zet jouw poppetje op START (let op de voorkant van het poppetje).
Laat jouw poppetje met behulp van de groene codes door het doolhof heenlopen, net zolang deze bij het einde (=END) is uitgekomen.
Let op!
De codes die je gebruikt hebt moet je in de goede volgorde hieronder opschrijven. Je mag de codes afkorten door alleen de gemarkeerde letters te noteren!

Invulblad:

Code 1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	
Code 7	
Code 8	
Code 9	
Code 10	
Code 11	
Code 12	
Code 13	
Code 14	
Code 15	
Code 16	
Code 17	
Code 18	
Code 19	
Code 20	

Opdracht 2:

Benodigdheden:

- 1 poppetje/pion
- Blad met codeermogelijkheden
- Doolhof nummer 2
- Deze opdrachtkaart met daarbij het invulblad

De opdracht:
Je gaat nu weer jouw poppetje door het doolhof laten lopen, maar dit keer maak je gebruik van zowel de groene codes als de rode code. Daarbij probeer je zo min mogelijk codes op te schrijven!
Bij de rode code mag je in plaats van de -------- zelf een getal invullen.

De codes die je gebruikt hebt schrijf je weer in de goede volgorde hieronder op. Als je de rode en de groene codes wilt combineren dat zet je deze achter elkaar: dus eerst de rode code en dan de groene code (bijvoorbeeld VOOR 3 STAPPEN – GA VOORUIT).

Invulblad:

Code 1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	
Code 7	
Code 8	
Code 9	
Code 10	
Code 11	
Code 12	
Code 13	
Code 14	
Code 15	
Code 16	
Code 17	
Code 18	
Code 19	
Code 20	

Opdracht 3A:

Benodigdheden:

- 1 poppetje/pion
- Blad met codeermogelijkheden
- Doolhof nummer 1
- Deze opdrachtkaart met daarbij het invulblad

De opdracht:
Je gaat nu weer jouw poppetje door het doolhof laten lopen, maar dit keer maak je gebruik van de groene, blauwe en zwarte codes. Jouw poppetje hoeft hierdoor niet de kortste weg te lopen.

De codes die je gebruikt hebt schrijf je weer in de goede volgorde hieronder op. Je moet ook hier weer codes combineren, daarbij is de volgorde: blauw, zwart, zwart en groen (bijvoorbeeld ALS – ER EEN MUUR – LINKS VAN MIJ STAAT – DRAAI RECHTSOM). Je mag ook meerdere groene codes als laatste combineren.

Invulblad:

Code 1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	
Code 7	
Code 8	
Code 9	
Code 10	
Code 11	
Code 12	
Code 13	
Code 14	
Code 15	
Code 16	
Code 17	
Code 18	
Code 19	
Code 20	

Opdracht 3B:

Benodigdheden:

- 1 poppetje/pion
- Blad met codeermogelijkheden
- Doolhof nummer 1 en 2
- Deze opdrachtkaart met daarbij het invulblad

De opdracht:
Je gaat nu weer jouw poppetje door het doolhof laten lopen met behulp van de groene, blauwe en zwarte codes. Daarbij is het de bedoeling dat met jouw serie van codes het poppetje zowel door doolhof 1 als door doolhof 2 kan komen.

De codes die je gebruikt hebt schrijf je weer in de goede volgorde hieronder op.

Invulblad:

Code 1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	
Code 7	
Code 8	
Code 9	
Code 10	
Code 11	
Code 12	
Code 13	
Code 14	
Code 15	
Code 16	
Code 17	
Code 18	
Code 19	
Code 20	

Opdracht 4:

Benodigdheden:

- 1 poppetje/pion
- 1 dobbelsteen
- Blad met codeermogelijkheden
- Doolhof nummer 3
- Deze opdrachtkaart met daarbij het invulblad

De opdracht:

Je gaat nu de serie van codes die je hierboven hebt opgeschreven uitproberen bij doolhof nummer 3. Er zijn nu twee mogelijkheden:

1. Jouw serie codes klopt en het poppetje bereikt het einde van het doolhof, je hoeft niets meer op te schrijven
2. Jouw serie codes klopt niet, je gaat de serie herschrijven hieronder zodat deze wel klopt

Als je jouw serie moet herschrijven mag je daarbij gebruik maken van de groene, blauwe, zwarte en paarse codes.

Invulblad:

Code 1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	
Code 7	
Code 8	
Code 9	
Code 10	
Code 11	
Code 12	
Code 13	
Code 14	
Code 15	
Code 16	
Code 17	
Code 18	
Code 19	
Code 20	

Opdracht 5:

De opdracht:
Een leerling kreeg dezelfde opdracht als jij bij opdracht 1 (alleen dan met doolhof nummer 3) en schreef onderstaande codes op als antwoord. Dit antwoord klopt echter niet. Kun jij de fout ontdekken? Verbeter de fout en schrijf de juiste code(s) in de laatste kolom op.

Invulblad:

Code 1	GV	
Code 2	GV	
Code 3	GV	
Code 4	DL	
Code 5	GV	
Code 6	GV	
Code 7	GV	

Opdracht 6:

De opdracht:
Een leerling kreeg dezelfde opdracht als jij bij opdracht 2 (alleen dan met doolhof nummer 5) en schreef onderstaande codes op als antwoord. Dit antwoord klopt echter niet. Kun jij de fout ontdekken? Verbeter de fout en schrijf de juiste code(s) in de laatste kolom op.

Invulblad:

Code 1	V2S – GV	
Code 2	DR	
Code 3	V2S – GV	
Code 4	DL	
Code 5	V2S – GV	
Code 6	DR	
Code 7	V3S – GV	
Code 8	DL	
Code 9	V3S – GV	
Code 10	DL	
Code 11	V4S – GV	
Code 12	DR	
Code 13	GV	

Opdracht 7A:

De opdracht:
Een leerling kreeg dezelfde opdracht als jij bij opdracht 3A (alleen dan met doolhof nummer 4) en schreef onderstaande codes op als antwoord. Dit antwoord klopt echter niet. Kun jij de fout ontdekken? Verbeter de fout en schrijf de juiste code(s) in de laatste kolom op.

Invulblad:

Code 1	A – EEM – VMS - GV	
Code 2		

Opdracht 7B:

De opdracht:
Een leerling kreeg dezelfde opdracht als jij bij opdracht 3B (alleen dan met doolhof nummer 4 en 5) en schreef onderstaande codes op als antwoord. Dit antwoord klopt echter niet. Kun jij de fout ontdekken? Verbeter de fout en schrijf de juiste code(s) in de laatste kolom op.

Invulblad:

Code 1	A – EGM – VMS - GV	
Code 2	AA – EGM – RVMS - DR	
Code 3	AA – EEM – LVMS – DL	
Code 4	AA – EEM – LRVMS – DL - DL	

Opdracht 8:

De opdracht:
Een leerling kreeg dezelfde opdracht als jij bij opdracht 4 en schreef onderstaande codes op als antwoord. Dit antwoord klopt echter niet. Kun jij de fout ontdekken? Verbeter de fout en schrijf de juiste code(s) in de laatste kolom op.

Invulblad:

Code 1	A – EGM – VMS - GV	
Code 2	AA – EGM – RVMS - DR	
Code 3	AA – EGM – LVMS – DL	
Code 4	AA – EEM – LRVMS – DL - DL	
Code 5		

Bijlage 2: Codes bij test Computational Thinking

...ga vooruit	draai linksom
	draai rechtsom
voor ----- stappen...	
als...	...er een muur...
...voor mij staat	...er geen muur...
...rechts van mij staat	...links, rechts en voor mij staat
...links van mij staat	. en de dobbelsteen hoger is dan 3...
anders...	anders als...

Bijlage 3: Verdeling van de leerlingen over de twee experimentele condities[3]

Groepsnummer	Conditie	Aantal totaal	Aantal jongens	Aantal meisjes	Aantal hoge scores Cito	Aantal lage scores Cito
1	Unplugged	9	5	4	6	3
	Plugged	9	5	4	5	4
2	Unplugged	13	5	8	7	6
	Plugged	13	6	7	9	4
3	Unplugged	12	6	6	8	4
	Plugged	10	4	6	7	3
4	Unplugged	6	4	2	6	0
	Plugged	6	4	2	5	1
5	Unplugged	13	5	8	9	4
	Plugged	13	6	7	7	6
6	Unplugged	13	5	8	11	2
	Plugged	13	6	7	12	1
7	Unplugged	11	5	6	8	3
	Plugged	11	5	6	7	4
8	Unplugged	9	4	5	8	1
	Plugged	8	3	5	8	0
9	Unplugged	4	2	2	1	3
	Plugged	4	2	2	2	2
10	Unplugged	6	4	2	3	3
	Plugged	6	3	3	4	2
11	Unplugged	4	2	2	1	3
	Plugged	4	2	2	1	3
12	Unplugged	8	5	3	2	6
	Plugged	7	4	3	3	5
13	Unplugged	9	5	4	6	3
	Plugged	8	5	3	7	1
14	Unplugged	10	4	6	8	2
	Plugged	10	5	5	7	3
15	Unplugged	9	4	5	1	8
	Plugged	10	4	6	2	8
Totaal	Unplugged	136	65	71	85	51
	Plugged	132	64	68	86	46

[3] De tabel dient als volgt gelezen te worden: per deelnemende klas (= groepsnummer) staat in de tabel vermeld hoeveel leerlingen deelgenomen hebben in de unplugged-conditie en in de plugged-conditie. Tevens is weergegeven hoeveel jongens en meisjes in beide condities hebben deelgenomen en hoeveel hoog en laag scorende leerlingen hebben deelgenomen.

Bijlage 4: Vragenlijst SAFL-Q (vertaling vanuit het Engels)

Deze vragenlijst gaat over de manier waarop je door de leerkracht tijdens de programmeerlessen begeleid werd en hoe jij dat ervaren hebt.
Hierover volgen 28 uitspraken.
Lees de zinnen goed en geef voor elke uitspraak aan in hoeverre je het er wel of niet mee eens bent. Je kunt kiezen uit:

1. helemaal oneens
2. oneens
3. eens/oneens
4. eens
5. helemaal eens

Zet achter iedere uitspraak een kruisje in jouw gekozen hokje.
LET OP! Het gaat steeds over jij het ervaart, niet over hoe jij denkt dat het zou moeten zijn. Er zijn geen ‘goede’ of ‘slechte’ antwoorden!

	1	2	3	4	5
1.De juf/meester stimuleert mij om na te denken over hoe ik mijn schoolwerk kan verbeteren.					
2. Nadat de juf/meester mijn werk heeft nagekeken bespreken we samen mijn antwoorden.					
3. Tijdens het maken van mijn schoolwerk vraagt de juf/meester hoe ik vind dat het tot nu toe gaat.					
4. De juf/meester laat mij meedenken over de manier waarop ik wil leren op school.					
5. Ik krijg van de juf/meester de mogelijkheid om zelf te bepalen wat mijn leerpunten zijn.					
6. Mijn juf/meester vraagt mij wat ik goed en minder goed heb gedaan in mijn schoolwerk.					
7. De juf/meester stimuleert mij om terug te kijken op mijn leerproces en om te bedenken wat ik een volgende keer anders kan doen.					
8. De juf/meester geeft mijn sterke punten aan op het gebied van leren.					
9. De juf/meester geeft mijn zwakke punten aan op het gebied van leren.					
10. Ik word aangemoedigd door mijn juf/meester om mijn leerproces te verbeteren.					
11. Ik krijg aanwijzingen van mijn juf/meester die mij helpen bij het leren.					
12. Mijn juf/meester bespreekt mijn gemaakte werk met mij zodat ik de lesstof beter begrijp.					
13. Mijn juf/meester bespreekt met mij mijn vorderingen.					
14. Na een opdracht laat mijn juf/meester mij weten hoe ik het de volgende keer beter kan doen.					
15. Mijn juf/meester bespreekt met mij hoe ik mijn sterke kanten kan gebruiken om mijn werk te verbeteren.					
16. Samen met mijn juf/meester bedenk ik een manier om mijn zwakke punten te verbeteren.					

	1	2	3	4	5
17. Als ik de uitleg niet begrijp dan probeert de juf/meester het op een andere manier aan mij uit te leggen.					
18. De juf/meester geeft mij aanwijzingen die mij helpen om de lesstof te begrijpen.					
19. Tijdens de les kan ik laten zien wat ik heb geleerd.					
20. De juf/meester stelt vragen die ik begrijp.					
21. De vragen van de docent helpen mij de lesstof te begrijpen.					
22. De juf/meester staat open voor mijn inbreng in de klas.					
23. Ik heb de mogelijkheid om vragen te stellen aan medeleerlingen over de les.					
24. Ik weet aan welke punten ik moet werken om mijn resultaten te verbeteren.					
25. Er is mogelijkheid om vragen te stellen.					
26. Ik weet aan welke eisen mijn werk moet voldoen.					
27. Als ik een opdracht krijg is het duidelijk wat ik hiervan kan leren.					
28. Met mijn werk laat ik zien wat ik kan.					

Bijlage 5: Vragenlijst TAFL-Q

Op de volgende 3 pagina's vindt u 28 stellingen.
Geef per stelling aan in hoeverre dit op u van toepassing is in uw eigen (reguliere) lessen.
De antwoorden staan in een 5-punts schaal van 1 tot 5, waarbij

1. helemaal oneens
2. oneens
3. eens/oneens
4. eens
5. helemaal eens

In hoeverre zijn de volgende uitspraken op u van toepassing?

	1	2	3	4	5
1.Ik stimuleer mijn leerlingen om na te denken over hoe zij hun schoolwerk kunnen verbeteren					
2. Na het nakijken van een toets bespreek ik met iedere leerling zijn of haar antwoorden					
3. Tijdens het maken van het schoolwerk vraag ik de leerlingen hoe ze vinden dat het tot nu toe gaat					
4. Ik laat mijn leerlingen meedenken over de manier waarop ze willen leren op school					
5. Ik geef leerlingen de gelegenheid zelf hun leerpunten te bepalen					
6. Ik vraag mijn leerlingen om aan te geven wat ze goed en minder goed hebben gedaan in hun schoolwerk					
7. Ik stimuleer leerlingen om terug te kijken op hun leerproces en om te bedenken wat ze een volgende keer beter anders kunnen doen					
8. Ik laat mijn leerlingen weten wat hun sterke punten zijn op het gebied van leren					
9. Ik laat mijn leerlingen weten wat hun zwakke punten zijn op het gebied van leren					
10. Ik moedig mijn leerlingen aan om hun leerproces te verbeteren					
11. Ik geef mijn leerlingen aanwijzingen die hen helpen bij het leren					
12. Ik bespreek het gemaakte werk met mijn leerlingen zodat zij de lesstof b eter begrijpen					
13. Ik bespreek met mijn leerlingen hun vorderingen.					
14. Na een toetsmoment vertel ik mijn leerlingen hoe zij hun zwakke prestaties kunnen verbeteren.					
15. Ik bespreek met mijn leerlingen hoe zij hun sterke kanten kunnen gebruiken om hun werk te verbeteren.					
16. Ik stel samen met mijn leerlingen een strategie vast om hun zwakke punten te verbeteren					
17. Als ik merk dat leerlingen een onderdeel niet begrijpen pas ik mijn instructie aan					
18. Ik geef de leerlingen aanwijzingen om hen te helpen de lesstof te begrijpen					
19. Tijdens de les kunnen de leerlingen laten zien wat ze hebben geleerd					

	1	2	3	4	5
20. Door vragen te stellen tijdens de les help ik leerlingen om de lesstof te begrijpen					
21. Ik stel mijn vragen op een begrijpelijke manier					
22. Ik ga met mijn leerlingen in discussie over de antwoorden					
23. Ik sta open voor de inbreng van mijn leerlingen in de klas					
24. Ik zorg ervoor dat de leerling weet aan welke punten hij of zij moet werken om zijn of haar resultaten te verbeteren					
25. Ik geef leerlingen de mogelijkheid om vragen te stellen					
26. De leerling weet aan welke eisen zijn of haar werk moet voldoen					
27. Ik zorg ervoor dat de leerlingen weten wat ze kunnen leren van een opdracht					
28. Ik kan zien of de leerling het leerdoel heeft behaald door zijn werk					